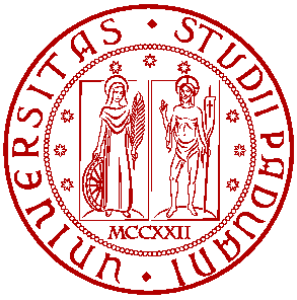


Università degli studi di Padova
Facoltà di Ingegneria

Graduation Thesis in

Ingegneria dell'Automazione

Distributed partitioning strategies in camera network application

candidate
Alberton Riccardo

advisor
Dr. Carli Ruggero

Academic Year 2010/2011

al nonno Marcello..

Contents

1 Introduction

In recent years systems of video surveillance composed by networks of cameras Pan-Tilt-Zoom (PTZ) have been largely deployed to monitor buildings and open environments. They cooperate in finding and tracking suspected objects or persons within cameras field of view. For this reason cameras are used as sensors, connected in a communication network, able to make decisions (analysing video data through properly designed algorithms).
In many applications a preliminary task the cameras are required to perform is the patrolling task. A good patrolling strategy minimize the time interval between two visits of the same point assuring that the whole area is covered.
A widely used approach to control these cameras is a centralized approach where a central unit manages all informations that come from cameras; usually in this context human presence is needed. Moreover this approach is based on global communication and optimization that does not scale well with the number of agents.
A distributed approach instead is based on the assumption that each camera (or agent) has an intelligence (i.e. has a microprocessor) so it can decide how to react based on what it sees and on informations that came from neighbours cameras. So control is performed just using local informations and without any human help. This is robust with the respect to agent failures or agent new appearance and it is usually preferred in large scale systems.
The partitioning problem is a key step that has to be make in patrolling strategy. It consists on assigning to each agent a portion of the environment thus dividing the workload among the components of the network. This decision is essential to improve the performance of the whole system and can be used to make the patrolling task easier.

The work is divided in two main parts. In the fist one we consider a one-dimensional scenario. Our main contribution consists on the design of a distributed partitioning policy able to reach an equitable partition (i.e., a partition where the workload is equally divided per each camera) in presence of physical constraints that limit the portion of the perimeter reachable to each camera. The demand for communication is very low. The algorithm requires just a pairwise gossip asymmetric communication, which is the more promising communication protocol for its potential industrialization. This problem is also solved using a centralized approach in order to verify through simulations that the optimal global partition is always reached. The problem of the fault of a camera is also taken into account.
In the second part we design heuristic distributed algorithms to achieve an equitable partition for a two dimensional environment. This problem is well known in literature. However algorithms designed are all suitable for a contin-

uous scenario and nothing has been proposed yet for a discrete environment. We propose a solution that exploits the discretization of the area. It not only reaches a global equitable partition but also provide 'fat' (i.e., as close as possible to a circle) subregions. Obtaining an equitable partition among all agents minimize the workload assigned to each camera. Obtaining subregions with a good shape improves overall performance (in particular reduces the time to reach every point of the subregion assigned to a camera). Two different version of this algorithm are designed. One requires at symmetric communication and the other one is able to achieve the goal just with asymmetric communication. The problem of the fault of camera is taken into account in both cases.

The presentation is organized as follows. In Section 2 we present briefly literature overview on these topics. In section 3 we deal with the problem of partitioning a perimeter. First we consider the centralized solution. Then we present the distributed solution. Comparisons are provided through extensive numerical simulations. In Section 5 we extended this problem to a two dimensional discrete environment. In Section 6 and 7 deep discussions based on numerical evidences are proposed. Finally, in the last section, we gather our conclusions and we propose some future directions for our work.

Literature overview. A broad discussion of partitioning and coverage control is presented in [3] which builds on the classic work of Lloyd [19] on algorithms for optimal quantizer design thorough "centering and partitioning". Since the beginning, coverage control algorithms have been applied to non-convex environments [14], [4], [10], unknown density function [13]. Applications on equitable partitioning on continuous environment can be found in [17], [18], [15]. Discrete environments are considered when dealing with Voronoi tasselations; for example in [5] there are distributed algorithms for a gossiping mobile agents.

The communication used, called *gossip communication model* is widely studied in the wireless communication literature; example references include [2], [9].

Coverage control and territory partitioning have applications in many fields. In cyber-physical systems, applications include automated environmental monitoring [6], fetching and delivery [21] and other vehicles routing scenarios [16]. Coverage of discrete sets is closely related to the literature on data clustering and k-means [11],[8], as well as the facility location or k-center problem [20]. Partitioning of graphs is also its own field of research, see [7]. Territory partitioning is also studied in models of competition between animal groups, see for example [1], [12].

2 Partitioning 1D

This section is organized as follows. First we formulate the problem we aim at solving including a brief summary of results obtained in previous works. Second we solve the problem in a centralized fashion by standard linear programming strategies. Third we introduce a novel algorithm which is fully distributed. Finally we report numerical simulations to compare our distributed solution with the centralized one.

2.1 Problem formulation

We are given a set of cameras with limited sensing and communication capabilities and a segment $\mathcal{L}$ to be patrolled. Let $L_{TOT} = |\mathcal{L}|$ be the length of the segment, n the number of cameras (or agents) and $|A_i| = [l_i, r_i]$ is the effective coverage of $i-th$ camera. $z_i(t) : \mathbb{R}_+ \to D_i$ denotes position of the f.o.v.(approximated to a point) of camera i as function of time t $v_i \in [-v_{i,max}, +v_{i,max}]$ is the speed of the $i-th$ camera during pan movement.
In this set up every camera can patrol just a portion of the whole segment $\mathcal{L}$, this physical constraint can be written as:

$$A_i \subseteq D_i\,, \qquad D_i = [D_{i,inf},\, D_{i,sup}] \tag{1}$$

for $i = 1, \ldots, N$. Note that in general D_i depends on the configuration of the $i-th$ camera and on the physical topology of the border.
Moreover we want all the segment $\mathcal{L}$ to be patrolled so we add a covering constraint:

$$\bigcup_{i=1}^{N} D_i = \mathcal{L} \tag{2}$$

A good patrolling strategy is one that minimize the time lag between two visits to the same location. Here is a formal definition of time lag.

Definition 1 (Time Lag) *For a given point x and a given time τ, we define* Time *Lag $\bar{t}(x,\tau)$ as the elapsed time from the most recent visit of $x \in \mathcal{L}$ by a camera (elapsed time from the last time t s.t. $\exists i \in 1, \ldots, N | z_i(t) = x$)*

$$\bar{t}(x,\tau) : \mathcal{L} \times \mathbb{R} \to \mathbb{R}^+$$

A uniformly "low" time lag over the segment $\mathcal{L}$ ensures that all locations are constantly monitored. It's clear that the less time pass between two visits the better it is, so we aim to minimize $\bar{t}$, constrained to the system dynamics

$$\dot{z}_i(t) = v_i(t)\,, \qquad \forall i s.t. \begin{cases} v_i(t) \in [-V_{i,max}, +V_{i,max}] \\ z_i(t) \in D_i \end{cases} \tag{3}$$

where the speed set v_i appears as a natural control for the system.

It is well known that an optimal solution without physical constraints ($D_i = \mathcal{L}$) is reached following those results:

Lemma 1 *If we have one camera monitoring a perimeter L_{tot} the optimal solution is reached by commanding a periodic motion with period $\bar{T}$ at maximum speed, obtaining $\bar{T} = \frac{2L_{tot}}{V_{max}}$*

Theorem 1 *Optimal coverage is attained assuming that every camera is moving at its maximum speed $|V_{i,max}|$ with a periodical motion of period $\bar{T}$ in non-overlapping coverage areas A_i^*:*

$$\bar{T} = \frac{2L_{tot}}{\sum_{i=1}^{N} V_{i,max}}\,, \qquad |A_i^*| = V_{i,max}\frac{\bar{T}}{2}\,.$$

So looking for the optimal solution implies looking for the optimal coverage areas A_i. This task is accomplished with distributed algorithms (without any central unit) and it is known how to the optimal solution is reached if we 'forget' physical constraints (or if the unconstrained solution is feasible $A_i^* \subseteq D_i\,, \quad i \in 1, \ldots, n$, in this case they are the same). Looking at communication protocols we can find three different scenarios:

- Synchronous: at each time every agent performs the communication with its neighbors and the update of its status
- Symmetric Gossip: at each iteration only a pair of neighbors camera communicate with each other and update their status
- Asymmetric Gossip: at each iteration there is only one camera transmitting to one of its neighbors (the only one that will update its status)

We now briefly show how the problem was solved in those three scenarios. We need to define

Definition 2 (Uniform Persistent Communication) *There is* Uniform Persistent Communication *if there exists $K \in \mathbb{N}$ s.t. for all t, any pair of neighboring cameras communicate with each other within the interval $[t, t + K]$.*

Definition 3 (Covering) *Given a segment $\mathcal{L}$ we define a* covering *as a collection $s = s_{i\,i=1}^{N}$ of N subintervals of $\mathcal{L}$ such that:*

1. $\bigcup_{i=1}^{N} s_i = \mathcal{L}$

With a synchronous communication, without physical constraints (i.e. $D_i = \mathcal{L}$) the algorithm is stated as follows.

At each time $t \in \mathbb{Z}_{\geq 0}$, each agent $i \in 1, \ldots, N$ keeps in memory limits of its segment $s_i(t)$ of $\mathcal{L}$, i.e. the two border points $l_i(t)$ and $r_i(t)$. The collection $s(0) = s_1(0), \ldots, s_N 0$ is a covering of $\mathcal{L}$. At each time $t \in \mathbb{Z}_{\geq 0}$ each agent i perform the following task:

1. agent i transmits to agent $i+1$ and $i-1$ its border points $l_i(t), r_i(t)$
2. $l_1(t+1) = l_1(t), \quad r_N(t+1) = r_N(t)$
3. $r_i(t+1) = \frac{r_{i+1}(t)v_i + l_i(t)v_{i+1}}{v_i + v_{i+1}}, \quad i = 1, \ldots, N-1$
4. $l_i(t+1) = \frac{r_i(t)v_i + l_{i-1}(t)v_{i-1}}{v_i + v_{i-1}}, \quad i = 2, \ldots, N$

In this case Covering Constraints (2) are satisfied and also the optimal partition is reached[1]. The updating of extremes is done to equalize the time required by agent i to travel from $r_i(t+1)$ to $l_i(t)$ at the speed v_i and the time required by the agent $i+1$ to travel from $r_{i+1}(t)$ to $l_{i+1}(t+1)$ at the speed v_{i+1}. This is called Neighbors'Equal-time travelling criterion.

With a symmetric communication, under the assumption of Uniform Persistent Communication, relaxing physical constraint the algorithm is:

At each time $t \in \mathbb{Z}_{\geq 0}$, each agent $i \in 1, \ldots, N$ keeps in memory limits of its segment $s_i(t)$ of $\mathcal{L}$, i.e. the two border points $l_i(t)$ and $r_i(t)$. The collection $s(0) = s_1(0), \ldots, s_N(0)$ is a covering of $\mathcal{L}$. At each time $t \in \mathbb{Z}_{\geq 0}$ only agents i and $i+1$ communicate with each other, where $i \in 1, \ldots, N-1$ is selected by a deterministic or stochastic process to be determined. Every agent $k \notin i, i+1$ sets $s_k(t+1) = s_k(t)$, while agents i and $i+1$ perform the following tasks:

1. agent i transmits to agent $i+1$ its subset $p_i(t)$ and vice-versa
2. $r_i(t+1) = l_{i+1}(t+1) = \frac{r_{i+1}(t)v_{i+1} + l_i(t)v_i}{v_i + v_{i+1}}$

Also in this case border points are updated according to the Neiqhbors'Equal-time criterion and the Covering Constraints are satisfied and the asymptotic convergence to the optimal solution is guaranteed as in the previous case by Consensus Theorem (but now P is stochastic).

In asymmetric scenario, under the assumption of Uniform Persistent Communication, the algorithm is stated as follows

[1] Observe that with the change of variable $x_i(t) := \frac{r_i(t) - l_i(t)}{v_i(t)}$ we can write system dynamic as $x(t+1) = Px(t)$, where P is pseudo-stochastic and asymptotic convergence $x_i(t) \to \frac{\bar{T}}{2}$ to optimal solution is guaranteed by Consensus Theorem

At each time $t \in \mathbb{Z}_{\geq 0}$, each agent $i \in 1, \ldots, N$ keeps in memory limits of its segment $s_i(t)$ of $\mathcal{L}$, i.e. the two border points $l_i(t)$ and $r_i(t)$. The collection $s(0) = s_1(0), \ldots, s_N 0$ is a covering of $\mathcal{L}$. At each time $t \in \mathbb{Z}_{\geq 0}$ agent i receive information from agent $i+1$ or $i-1$, where $i \in 1, \ldots, N$ is selected by a deterministic or stochastic process to be determined. Every agent $k \neq i$ sets $s_k(t+1) = s_k(t)$, while agents i perform the following tasks:

1. **if** i receives from $i+1$ **then**
2. computes $r^* = \frac{r_{i+1}(t)v_i + l_i(t)v_{i+1}}{v_i + v_{i+1}}$
3. **if** $r^* > l_{i+1}$ **then**
4. $r_i(t+1) = r^*$
5. **else**
6. $r_i(t+1) = l_{i+1}(t)$
7. **end**
8. **else** % i receives from $i-1$
9. computes $l^* = \frac{l_{i-1}(t)v_{i-1} + r_i(t)v_i(t)}{v_{i-1} + v_i}$
10. **if** $l^* < r_{i-1}(t)$ **then**
11. $l_i(t+1) = l^*$
12. **else**
13. $l_i(t+1) = r_{i-1}(t)$
14. **end**
15. **end**

Note that in this scenario the covering constraint might be violated if we keep always r^* (or l^*) chosen according to the Neighbors Equal Time Criterion. In particular it becomes active when $r^* < l_{i+1}(t)$ (or $l^* > r_{i-1}(t)$) and so the choice is forced to be $r^* = l_{i+1}(t)$ (or $l^* = r_{i-1}(t)$) to keep all the segment covered.
In the next section we propose a solution to solve this problem with asynchronous protocol adding also physical constraints. It will be performed using a centralized and then a distributed approach.

2.2 A centralized solution

We propose in this section the solution of the problem encountered using a centralized approach. In particular the problem is stated as linear programming problem where the linear constraints are the limits given by physical constraint.
The segment to be patrolled is $\mathcal{L} = [-L, +L]$ and we want to find positions $\mathbf{x} := x_1, \ldots, x_{N-1} \in \mathcal{L}$ of camera's border that equalize as more as possible

all of the areas assigned to each camera. We define $f_i(\mathbf{x})$ as the portion of $\mathcal{L}$ assigned to the i-th camera. For N cameras we have $N-1$ partitions and we can write:

$$f_1(\mathbf{x}) = \frac{x_1 + L}{v_1} \tag{4a}$$
$$f_2(\mathbf{x}) = \frac{x_2 - x_1}{v_2} \tag{4b}$$
$$\vdots \tag{4c}$$
$$f_{N-2}(\mathbf{x}) = \frac{x_{N-2} - x_{N-3}}{v_{N-2}} \tag{4d}$$
$$f_{N-1}(\mathbf{x}) = \frac{L - x_{N-2}}{N-1} \tag{4e}$$

We can write the unconstrained problem as

$$z = \min_{x_1,\dots,x_n} \max_{i=1,\dots,N-1} f_i(\mathbf{x}) \tag{5}$$

thus our aim is to minimize the biggest segment $f_i(\mathbf{x})$. We have to now to constrain points x_i to stay inside the overlapping parts of physical constraint of adjacent cameras. Calling this part $\bar{D}_i := D_i \cap D_{i+1} = [\bar{D}_{i,inf}, \bar{D}_{i,sup}]$ we have $\bar{D}_i \neq \emptyset$ due to the covering constraint (2) and with $x_i \in \bar{D}_i$ for all i we clearly satisfy constraints (2) and (1). Calling $\varepsilon := \max f_i(x)$ we can write (5) as:

$$\begin{aligned} \min \varepsilon \\ f_1(\mathbf{x}) \leq \varepsilon \\ \vdots \\ f_{N-1}(\mathbf{x}) \leq \varepsilon \\ x_1 \in \bar{D}_1 \\ \vdots \\ x_{N-1} \in \bar{D}_{N-1} \end{aligned} \tag{6}$$

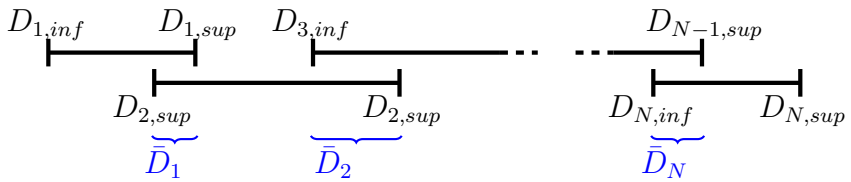
Finally we can write it in the standard form defining $\bar{\mathbf{x}} := [\varepsilon, x_1, \dots, x_N]^T$ and $\mathbf{c}^T = [1, 0, \dots, 0]$,

$$\begin{aligned} \min \mathbf{c}^T \bar{\mathbf{x}} \\ -\varepsilon + x_1 &\leq -L \\ -\varepsilon - x_1 + x_2 &\leq 0 \\ &\vdots \\ -\varepsilon - x_{N-2} + x_{N-1} &\leq -L \\ x_1 &\geq \bar{D}_{1,inf} \\ x_1 &\leq \bar{D}_{1,sup} \\ &\vdots \\ x_{N-1} &\geq \bar{D}_{N-1,inf} \\ x_{N-1} &\leq \bar{D}_{N-1,sup} \\ \varepsilon, x_1, \dots, x_{N-1} &\geq 0 \end{aligned} \tag{7}$$

In this form the problem can be easily solved using the simplex algorithm.

2.3 A distributed solution

We show now the distributed solution that we propose to solve the problem. As before $A_i = [l_i, r_i]$ is the patrolled segment, $A_i \subseteq D_i$, $D_i = [D_{i,inf}, D_{i,sup}]$ are physical constraints and $\bar{D}_i = D_i \cap D_{i+1} = [D_{i+1,inf}, D_{i,sup}]$. The protocol used is asynchronous, so as seen before, at each time $t \in \mathbb{Z}_{\geq 0}$ just one camera send information about its patrolled area and its maximum reachable area. We work under the assumption of Uniform Persistent Communication. Suppose that every camera is initialized to its physical constraint $A_i(0) = D_i$ for all i, and that $i+1$ send to i points $l_{i+1}(t), r_{i+1}(t)$. The case where $i-1$ sends informations to i is conceptually the same. In a general situation the two cameras could have different speed v_i and v_{i+1}.



Once received those points i update just $r_i(t)$ according to the law:

$$\frac{r_i(t+1) - l_i(t)}{v_i} = \frac{r_{i+1}(t) - r_i(t+1)}{v_{i+1}} \tag{8}$$

that gives:

$$r_i(t+1) = \frac{v_{i+1} l_i(t) + v_i r_{i+1}(t)}{v_i + v_{i+1}} = r^* \tag{9}$$

so we aim to equalize the time needed by cameras for going from one side to the other of their own region A_i. If all the cameras have the same speed we obtain: $r_i(t+1) = \frac{r_{i+1}(t)+l_i(t)}{2}$.
At this point there could be two problems. The first one is that covering constraints might be not satisfied, in fact if $r^* < l_{i+1}(t)$ there will be an uncovered area of length $s_{\text{uncov}} = l_{i+1} - r^*$. In this case r^* must be updated following the active constraint, we impose

$$\textbf{if } r^* < l_{i+1}(t) \quad \Longrightarrow \quad r_i(t+1) := l_{i+1}(t) \tag{10}$$

The second problem is that the physical constraint might be not satisfied, in fact in (9) r^* could go outside the boundaries of D_i, so if that happen we modify the law as:

$$\textbf{if } r^* > D_{i,sup} \quad \Longrightarrow \quad r_i(t+1) := D_{i,sup} \tag{11}$$

With this control law both covering and physical constraints are satisfied and we can summarize the algorithm as follow.

At each time $t \in \mathbb{Z}_{\geq 0}$, each agent $i \in 1, \ldots, N$ keeps in memory limits of its segment $A_i(t)$ of $\mathcal{L}$, i.e. the two border points $l_i(t)$ and $r_i(t)$. The collection $A(0) = A_1(0), \ldots, A_N(0)$ is a covering of $\mathcal{L}$. At each time $t \in \mathbb{Z}_{\geq 0}$ agent i receive information from agent $i+1$ or $i-1$, where $i \in 1, \ldots, N$ is selected by a deterministic or stochastic process to be determined. Every agent $k \neq i$ sets $A_k(t+1) = A_k(t)$, while agents i perform the following tasks:

1. **if** i receives from $i+1$ **then**
2. computes $r^* = \frac{r_{i+1}(t)v_i + l_i(t)v_{i+1}}{v_i + v_{i+1}}$
3. **if** $r^* < l_{i+1}(t)$ **then**
4. $r^* = l_{i+1}(t)$
5. **elseif** $r^* > D_{i,sup}$ **then**
6. $r^* = D_{i,sup}$
7. **end**
8. $r_i(t+1) = r^*$
9. **else** % i receives from $i-1$
10. computes $l^* = \frac{l_{i-1}(t)v_{i-1} + r_i(t)v_i}{v_{i-1} + v_i}$
11. **if** $l^* > r_{i-1}(t)$ **then**
12. $l^* = r_{i-1}(t)$
13. **elseif** $l^* < D_{i,inf}$ **then**

14. $l^* = D_{i,inf}$
15. **end**
16. $l_i(t+1) = l^*$
17. **end**

3 Simulation results for 1D case

In this section we report simulative results obtained running the distributed algorithm developed. Firstly we provide details of some specific aspects that can occur showing algorithm responses. Then we find meaningful parameters in order to give a global overview of the procedure developed and its convergence. Comparison with results obtained by simplex algorithm are reported in some instances. Without loss of generality we will assume that every camera has the same constant speed $v_i = V, i \in \{1, \dots, N\}$. All the code is developed using MATLAB software.

Starting from balanced initialization ($A_i^* \subseteq D_i, \in 1, \dots, N$) we show that, as we expected, the distributed algorithm reaches the optimal solution where each segment has the same length of the others. We take as instance $n = 5$ cameras and a perimeter of length $\mathcal{L} = 50$. We can see this starting point in Figure 1, where each coloured box represents A_i, the part of perimeter assigned to i-th camera. Under the perimeter we highlighted intervals $I_i = [D_{inf,i+1}\, D_{sup,i}], i \in \{2, \dots, N-1\}$ so we can rewrite physical and covering constraints as

$$r_i \in I_i, \quad l_{i+1} \in I_i, \quad l_1 = 0, \quad r_N = |\mathcal{L}|, \quad i \in \{1, \dots, N\}. \tag{12}$$

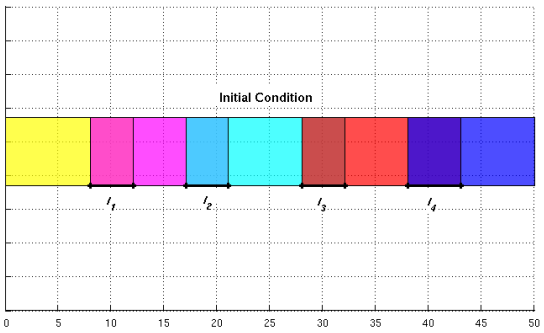


Figure 1: Initial condition of patrolling cameras

We report the evolution of the algorithm for some instants. Some observation can be made looking on Figure 2. First of all we see that optimal partition is reached, for $t = 1000$ we have $|A_i| = |A_j|, i \neq j, i, j \in \{1, \dots, 5\}$ as we expected when physical constraints do not became active the algorithm works as in the previous case reaching the optimal solution. Observe that just few iterations are needed to reach a good partition, i.e. if we are interested in

a sub-optimal results, fixing $\varepsilon > 0$ and asking that $max||A_i| - |A_j|| < \varepsilon, i \neq j$ a solution is found with much fewer iterations (of course it will depend also on the number of agents, and it will increase as N increase...). In Figure 3 is possible to see how the maximum partition decrease toward the optimal length $\frac{\mathcal{L}}{N} = 10$.

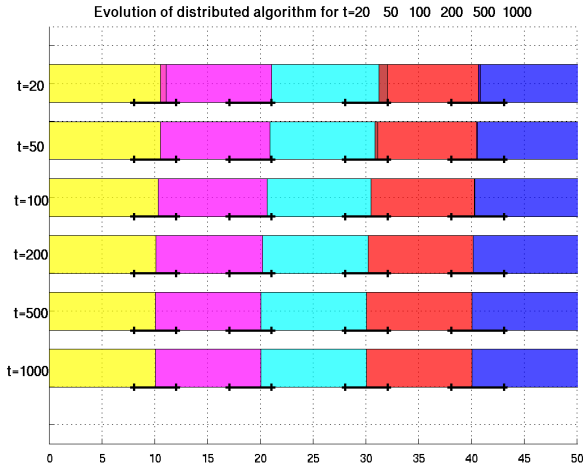


Figure 2: Evolution of partitions of $\mathcal{L}$

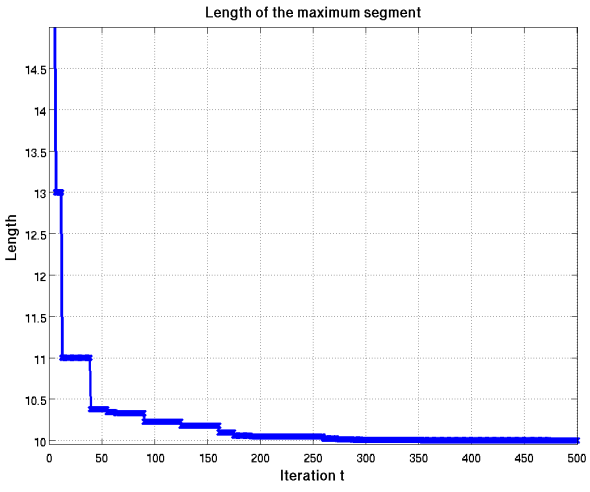


Figure 3: Trend of length of the longest region to be patrolled

In the next example we will see the behaviour of the algorithm when some constraints became active, when some camera is not able to patrol the whole

zone optimal for the unconstrained solution. To show this we can take the previous example and modify a little bit the starting point. Suppose that, for some reason, the last camera is not able to patrol the whole area A_5^* because its limits are smaller. For instance we can take $D_5 = [43\ 50]$ that has a length of $7(< \frac{\mathcal{L}}{N})$. The situation appears in Figure 4

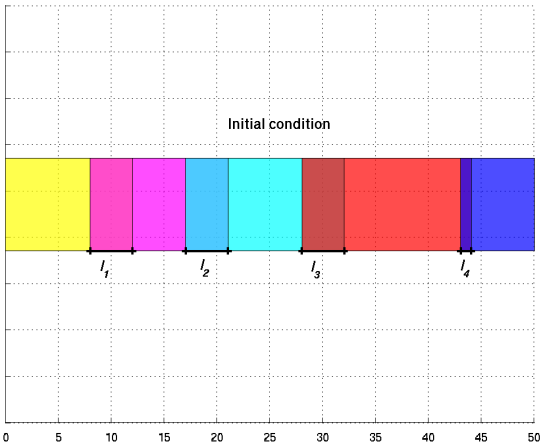


Figure 4: An initial condition where camera there is $D_5 < \frac{\mathcal{L}}{N}$

If we look at the evolution when rounds of communication involve the fourth and fifth cameras we encounter a situation like the one depicted in Figure 5. When the last camera receives informations from its previous camera 4 (that's the meaning of $5 \to 4$ in figure), if it would update the left extreme l_5 as in (9) obtaining, for the left extreme

$$l_5(t+1) = \frac{l_4(t) + r_5(t)}{2} = \frac{28 + 50}{2} = 39$$

but the last camera can not patrol perimeter behind the length 42. In this case the best the algorithm can do is to put $l_5(t+1) = D_{5,inf}$ thus leaving the state unchanged according to the camera's constraints.

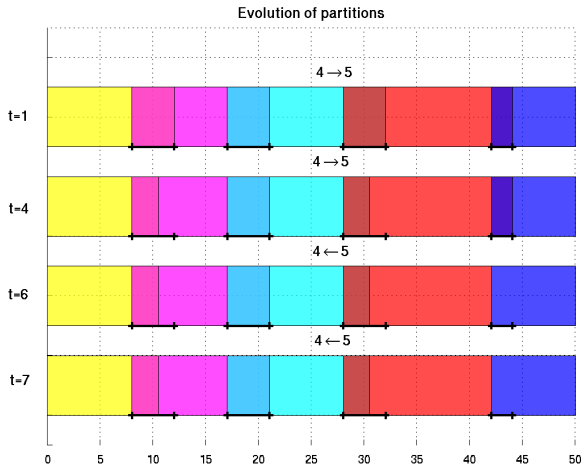


Figure 5: An initial condition where camera there is $D_5 < \frac{\mathcal{L}}{N}$

For $t = 4$ the state is the same and no update is made. For $t = 5$ communication involves other cameras and for $t = 6$ there is camera 4 receiving from camera 5 ($4 \leftarrow 5$) and an updating is needed for $r_4(t)$. If we would follow the law (9) we will obtain as before

$$r_4(t+1) = \frac{28+50}{2} = 39,$$

but updating the state in this way means leaving space not patrolled and this is not allowed by the covering constrain. So the update is modified as $r_4(t+1) = D_{5,inf}$ and that is the best it is possible to do. In the next iteration, namely for $t = 7$, the situation is repeated as before and no changes can be made.
In order to see at what partition the algorithm converges we can see Figure 6, where we compare with initial conditions and simplex solution. The distributed one is taken at a time $t = 1000$ that is arbitrary big to assure this is final result. It's evident that the two solution are exactly the same, physical and covering constrains are satisfied (as we seen before). In both cases an equitable partition is reached "forgetting" the contribute of the last camera, constrained to be long just 8 (instead of 10) from point 42 to 50. To cover all the perimeter $\mathcal{L}$ length of remaining cameras increase to the value 10.5, so we have not only reached the minimum length for the longest segment,according to constraints, as we can see in Figure 7 but also forced the others to be of the same size.

The previous example is the evidence that the two approaches are the same not only for the longest segment but also for the others. This may

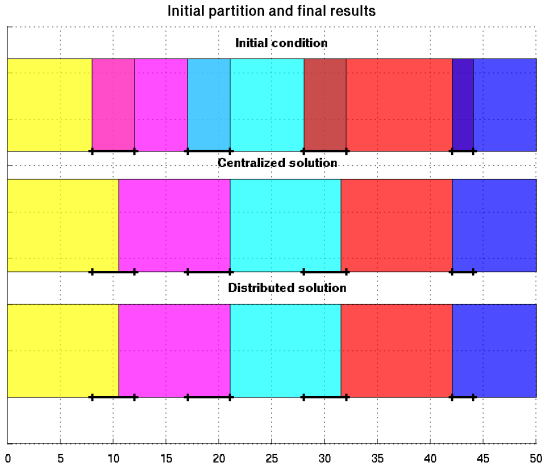


Figure 6: Initial partition and final result obtained with a distributed and centralized approach

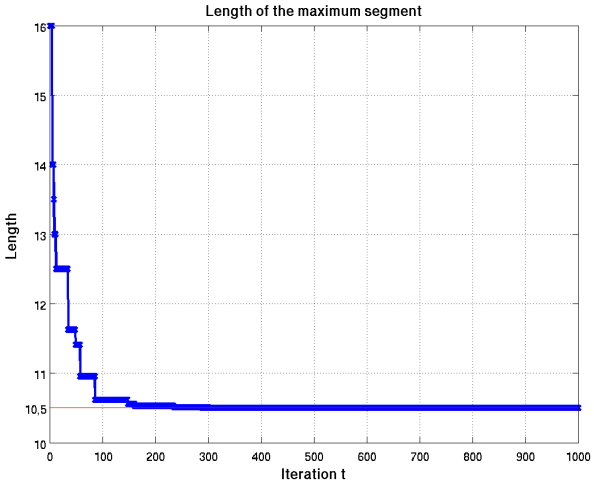


Figure 7: Length of the longest partition during iterations

	l_1	$r_1 = l_2$	$r_2 = l_3$	$r_3 = l_4$	$r_4 = l_5$	$r_5 = l_6$	r_6
Centralized	0	17.51	33.53	51	70	81.36	88
Distributed	0	17	34	51	70	79	88

Table 1: Details of extremes of partitions in Figure 8

seem quite strange thinking that in (6) we aim to minimize just the biggest subinterval A_i. In fact this is what happen and the previous behaviour is not the rule. The example shown in Figure 8 has different initial conditions and a different number of cameras. In this case both strategies give the same solution for the biggest segment $A_4 = [51\ 70]$ but are different for the shorter ones, for instance with simplex we obtain $A_5 = [70\ 81.36], A_6 = [81.36\ 88]$ and with the distributed $A_5 = [70\ 79], A_6 = [79\ 88]$.

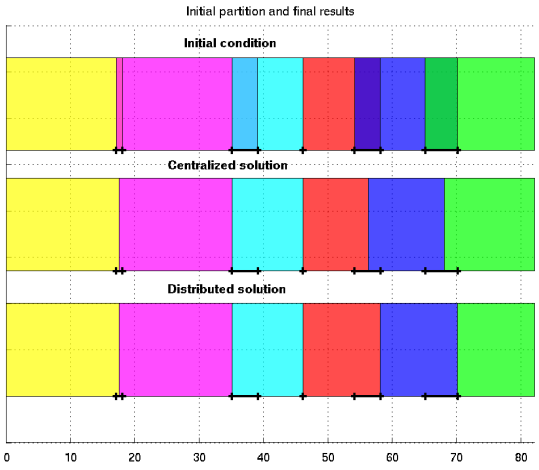


Figure 8: Length of the longest partition during iterations

These results confirm the intuition that our algorithm between two fixed points always reach an equitable partition. In this case the fixed points are 0 and the first physical constrain active, 51 and to each camera is assigned a segment long 17. Other two fixed points are the second physical constrain active and the end of $\mathcal{L}$, in this case we have two agents and the portion assigned to each of them is equal to 9. This is useful if we want not only minimize the longest part assigned to one camera (thus the bottleneck of the patrolling time) but also distribute uniformly the work assigned to others cameras.

Another case that has been taken into account is the case of a faulting

camera. An example of the behaviour of the algorithm is shown in Figure 9 where at iteration $t = 50$ the third camera stop working. As we can see, the algorithm cover (as long as it is possible) the subinterval left unpatrolled by the faulting camera. Moreover it distribute equally the additional workload among the remaining cameras of the network (see instant $t = 149$). As the faulting camera returns to work ($t = 150$) the partition change again and the optimal configuration is reached naturally, without any changing of the algorithm.

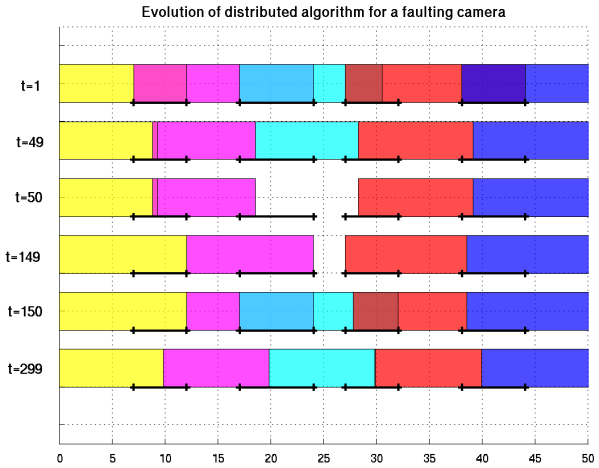


Figure 9: Evolution of the algorithm in case of the fault of a camera

A simulative demonstration of the correctness of the distributed solution can be obtained running a Monte Carlo test, we chose to run the test for $n = 1000$ times, focusing on the behaviour of the length of the longest portion ε of $\mathcal{L}$ we wanted to see if it is true that almost surely it's the same as the optimal solution founded with simplex. So we define $\lambda = |\varepsilon_{centr} - \varepsilon_{distr}|$ with obvious meaning if symbols. As we can see in Figure 10 the answer is definitely positive, being the difference to zero due just at numerical approximation. So looking at mean $\mathbb{E}[\lambda]$ and variance $\sigma^2(\lambda)$ of λ we obtain:

$$\mathbb{E}[\lambda] = 1.4218 \times 10^{-08}$$
$$\sigma^2(\lambda) = 6.7792 \times 10^{-14}$$

In the following Figure 10 we can see in the first box the position of ε_{centr} and ε_{distr} for every iteration, than we can see the difference between the two variables and its mean. Finally is shown the difference among the length of each segments assigned to cameras. As we observed before there is no reason

why they should converge to identical values. It follows Figure 11 that is a zoom of the previous figure.

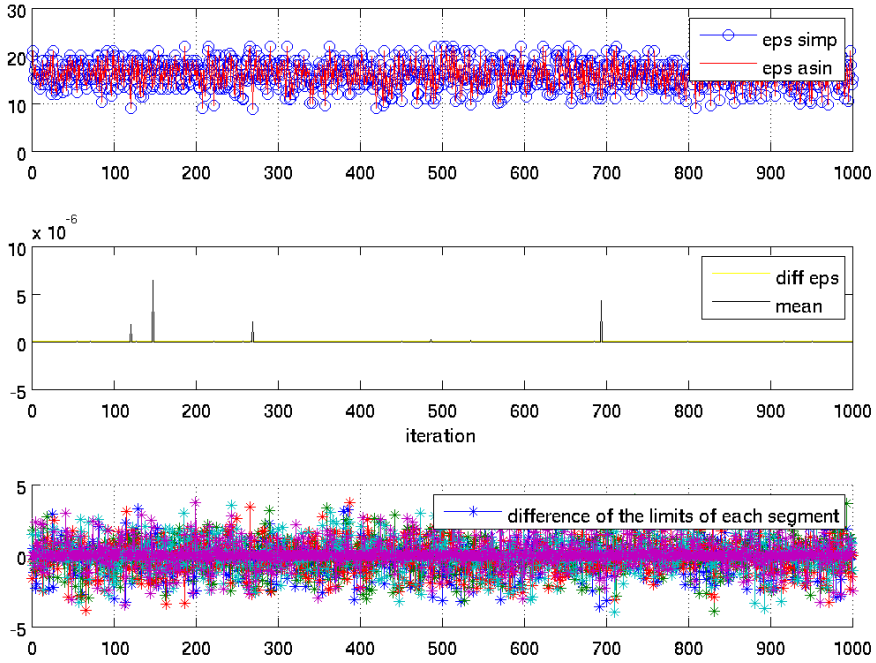


Figure 10: Statistic of the Monte Carlo test

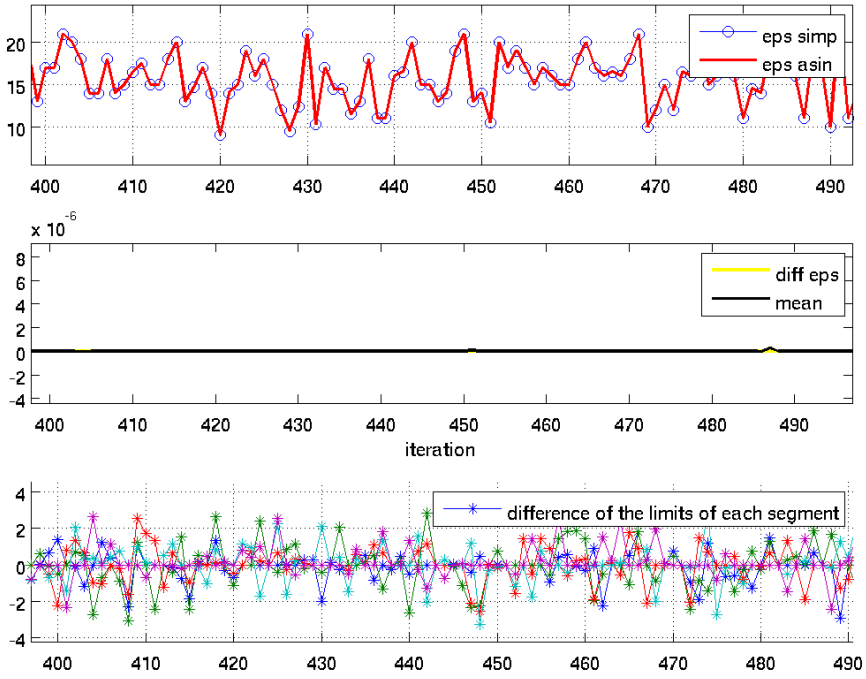


Figure 11: Zoom on statistic of the Monte Carlo test

4 Partitioning 2D

In this section we will deal with a generalization of the problem stated before considering instead of a perimeter two dimensional surface. We will find a partitioning policy, i.e. a distributed algorithm that equally (with respect to a measure) partitions the discretized environment. Each subregion will be assigned to a camera (or agent) whose task is to patrol it. Supposing that all cameras have same features, and working with a uniform environment, the solution of this problem will distribute equally the workload to each agent.

4.1 Problem formulation

Let the finite set $\mathcal{Q}$ be the discretized environment (of the bounded workspace A). We assume that the elements of $\mathcal{Q}$, which can be thought as location, are connected by weighted edges. In other words, we let $G = (Q, W, w)$ be an (undirected) weighted graph with edge set $E \subset Q \times Q$ and weight map $w : E \to \mathbb{R}_{>0}$; we let $w_e > 0$ be the weight of edge e. We assume that G is connected and think of the edge weights as distances between locations. In the following they will be all equals, in order to have a uniform discretization where all cells have the same size. An example is shown in Figure 12.

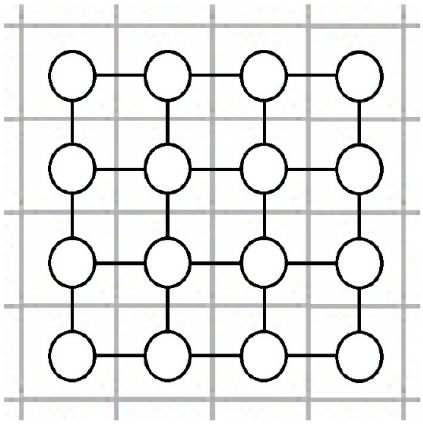


Figure 12: Example of a graph of a discretized area

In any weighted graph G there is a standard notation of distances between nodes defined as follows. A *path* in G is an ordered sequence of nodes such that any pair of consecutive nodes in the sequence is an edge of G. The *weight of a path* is the sum of the weights of all the edges in the path. Given two vertices h and k in G, the *distance* between h and k, denoted by $d_G(h, k)$,

is the smallest weight of any path from h to k, or $+\infty$ if there is no path from h to k. In other words, the distance between two nodes is the weight of the shortest path between them. By convention, $d_G(h,k) = 0$ if $h = k$. Note that $d_G(h,k) = d_G(k,h)$, for any $h, k \in Q$. If the graph is connected, then the distance between any two nodes is finite. Moreover if the environment is convex we will approximate $d_G(h,k)$ with the euclidean distance $d(h,k)$ computationally faster and sufficiently accurate for our purposes. The shortest path on a graph can be founded using Dijkstra's algorithm and it will be used just when strictly necessary.

Analogously, we define local distances on induced subgraphs of $G = (Q, E, w)$. Given $I \subset Q$, the *subgraph induced by restriction of G to I*, denoted by $G \cap I$, is the graph with the set of nodes equal to I and with the set of weighted edges containing all weighted edges of G where both vertices belong to I. In other words, we set $(Q, E, w) \cap I = (Q \cap I, E \cap (I \times I), w|_{I \times I})$. The induced subgraph is a weighted graph that is equipped with a notion of distance between nodes. Given $h, k \in I$,we write $d_I(h,k) := d_{G \cap I}(h,k)$. Note that $d_I(h,k) \geq d_G(h,k)$.

We remark that in this scenario a continuous environment is substituted by an *occupancy grid map* (in other words a discretization of the environment), where each grid cell is either a free space or an obstacle (i.e. it's occupied). We define two free cells as *adjacent* if they border each other in the grid map. The group of agents (robots or cameras) is then tasked with partitioning the graph of the free cells. Given the graph $G = (\mathcal{Q}, E, w)$, we define *connected subset of* $\mathcal{Q}$ as a subset $S \subset \mathcal{Q}$ such that $S \neq \emptyset$ and $G \cap S$ is connected. Moreover let $C(\mathcal{Q})$ denotes the set of such subsets. We can define partitions of Q into connected sets as follows.

Definition 4 (Connected partitions) *Given the graph $G = (\mathcal{Q}, E, w)$, we define a connected N-partition of $\mathcal{Q}$ as a collection $p = \{p_i\}_{i=1}^N$ of N-subsets of $\mathcal{Q}$ such that*

1. $\bigcup_{i=1}^N p_i = \mathcal{Q}$;
2. $p_i \cap p_j = \emptyset \; if \; i \neq j$;
3. $p_i \neq \emptyset \; for \; all \; i \in \{1, \ldots, N\}$;
4. $p_i \in C(\mathcal{Q}) \; for \; all \; i \in \{1, \ldots, N\}$.

Let P to be the set of such partitions.

A technical assumption is then needed to define the *generalized centroid* of connected subset. In what follows, we assume that a *total order* relation, $<$, is defined on $\mathcal{Q}$: hence, we can also denote $\mathcal{Q} = \{1, \ldots, |\mathcal{Q}|\}$.

Definition 5 (Centroid) *Let $\mathcal{Q}$ be a totally ordered set, and $R \in C(\mathcal{Q})$.*

We define the set of generalized centroids

$$C(R) := \arg\min_{h \in R} \left\{ \sum_{k \in R} d_R(h,k) \right\}, \tag{13}$$

and the map $\mathrm{Cd} : C(G) \to \mathcal{Q}$ *such that* $\mathrm{Cd}(R) := min\{c \in C(R)\}$. *We call* $Cd(R)$ *the* generalized centroid *of* R.

In subsequent use we will drop the world "generalized" for brevity. Note that with this definition the centroid is well defined, thanks to the ordering assumption, the centroid of region belongs to that region. It is a good approximation of the 'real' centroid of the region R and the use of Euclidean distance will not affects this definition.

In this scenario we will deal with partitioning $\mathcal{Q}$ into connected subsets that have the same number of elements. We can define these type partitions of $\mathcal{Q}$ as follows.

Definition 6 (Equitable partitions) *Given the graph* $G = (\mathcal{Q}, E, w)$, *we define an equitable* N*-partition of* $\mathcal{Q}$ *as a collection* $p = \{p_i\}_{i=1}^{N}$ *of* N*-subsets of* $\mathcal{Q}$ *such that*

1. p *is a connected partition*
2. $|p_i| = |p_j|\,, i \neq j\,, i, j \in \{1, \ldots, N\}$

So we want to find a distributed policy to divide $\mathcal{Q}$ in N subsets (or subregions of A) $R_1, \ldots, R_N$ that define an equitable partition $\{R_i\}_{i=1}^{N}$ of $\mathcal{Q}$. Each camera i is assigned to subregion R_i and we take as index of performance:

$$\varepsilon(t) = \max_{i \in \{1,\ldots,N\}} |R_i(t)| - \min_{i \in \{1,\ldots,N\}} |R_i(t)|\,. \tag{14}$$

When it becomes equal to zero all subregions have the same size. Observe that in general more than one equitable partition exists for a given workspace, for example see Figure 13, we aim to reach a partition with 'fat' (i.e. with small diameter) subregions. This because in general fat subregions improve overall performance (useful for the patrolling task, for example).

4.2 Algorithm proposed with symmetric communication

In this section we present the algorithm proposed to reach the equitable partition when agents have a symmetric communication protocol and the environment is discretized. The goal is to reach a partition where all subregions have the same size and with a regular and 'fat' shape. The idea of

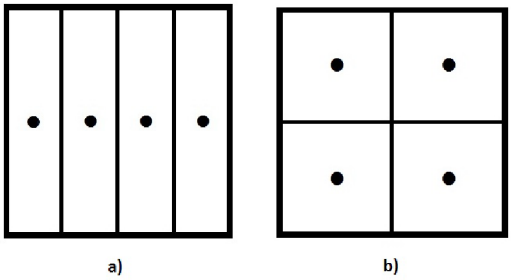


Figure 13: Two equitable partitions with different subregions shape

the algorithm is to select points to be exchanged in order to achieve these two goals at the same time. So at each round of communication cameras try to reduce the difference between their areas and try to do it keeping a borderline as smooth as possible. When cameras have the same area they will exchange points in order just to regularize the shape of their region.

4.2.1 Notations and euristic

Let be $R_i(t)$ and $R_j(t)$ the pair of communicating cameras in a round of communication at the time $t \in \mathbb{Z}$. The protocol of communication allows i and j to exchange information in a symmetric way, thus i send to j informations about region $R_i(t)$ and coordinates of centroid $\mathrm{Cd}_i(t)$ and camera j does the same to i (we denote $i \leftrightarrow j$). We define $\Gamma_{i,j}(t)$ as the set of points of $R_i(t)$ that are adjacent to $R_j(t)$ in the occupancy grid map at time t, we write $\Gamma_{i,j}(t) = \{h \in R_i(t) : h \in adj(R_j(t))$ where $adj(R_j(t)$ is the set of points adjacent to, at least, one point of $R_j(t)$. Analogously $\Gamma_{j,i}(t)$ is the set of point of $R_j(t)$ adjacent to $R_i(t)$.

In order to keep regular subregions we will *exchange* [2] just few cells at each step. Doing so under the Uniform Persistent Communication we induce little changes along all of the frontiers and, as the time pass, we obtain that each camera exchange points with all its neighbour [3]. Moreover we clearly want

[2]Saying that cameras i and j exchange a point v we mean that

$$\text{if } v \in R_i(t), v \notin R_j(t) \Rightarrow v \in R_j(t+1), v \notin R_i(t+1)$$
$$\text{else if } v \in R_j(t), v \notin R_i(t) \Rightarrow v \in R_i(t+1), v \notin R_j(t+1)$$

[3]The alternative of i and j exchanging all the possible points to reach the same area $|R_i(t+1)| = |R_j(t+1)|$ leads to get an equitable partition faster but without control of the shape of subregions. The final results (in term of shapes) will be highly sensitive to initial conditions. If we start with a balanced and regular initial partition we can hope

to have connected subregions at every step, so points that can be exchanged are just those points that belong to sets $\Gamma_{i,j}(t)$ and $\Gamma_{j,i}(t)$. In what follows we suppose that there is no overlapping between cameras. This will not affect the procedure because if at time $t \in \mathbb{Z}$ there is overlap, defined as

$$\Omega_{i,j}(t) = \{h \in \mathcal{Q} : h \in (R_i(t) \cap R_j(t))\} \neq \emptyset$$

it will be resolved immediately: each camera takes overlapping points that are closer to its centroid and release the others. By convection overlapping points equally distanced from the two centroids are assigned to the camera with lower number (in what follows we suppose $i < j$).

Now we need to know how many points to exchange. We said before that we want to exchange just few points, so we choose to exchange 2 points if the difference between the two areas is greater than 2 cells (i.e. $||R_i| - |R_j|| > 2$) and just one point if this difference is equal to 2. If the difference is less or equal than 1 we will focus just in regularizing the border.

We will deal now with the case $||R_i(t)| - |R_j(t)|| \geq 2$ and we suppose that, for example, $|R_i(t)| > |R_j(t)|$. This means that camera i has to give some cells to camera j to reduce the difference between the two subregions to be patrolled. The choice of two(or one) points to exchange inside $\Gamma_{i,j}(t)$ isn't unique and we can use this degree of freedom to assure that borders keep a regular shape. We need a criterion to detect the bests among them for this purpose.

The idea is to assign a *priority* to each point that quantify how much that cell is 'outside' its subregion. Thus the higher will be this number the more important will be that that the camera release this point. We define a map $\mathrm{pr} : \Gamma_{i,j}(t) \to \mathbb{R}$,

$$\mathrm{pr}(h, R_i(t)) = |adj(h)| + \frac{1}{2}|diag(h)|,$$

where $h \in \Gamma_{i,j}(t)$, $|adj(h)|$ gives the number of adjacent nodes to h that don't belongs to $R_i(t)$ and $|diag(h)|$ gives the number of nodes that don't belong to $R_i(t)$ and that are close to h on the diagonal direction. The last term $diag(h)$ is necessary to get the function $pr(h, R_i(t))$ more accurate on weighting nodes. Observe that it does not depend on the neighbor j chosen (there could be more than one for example on corners). As defined, priority gives us a notion on how much the point h is 'detached' from $R_i(t)$. An exception to this definition has to be made if h is on the border of the environment $\mathcal{Q}$. In this case a constant term 1 is added at the sum to compensate the fact that

of getting a good final results, but this is not true in general where cameras could be initialized roughly with big differences of areas to be patrolled with skinned shape.

those points have less adjacent nodes than the others. Note that without this adjustment they will be hardly exchangeable due to their intrinsic low priority. We can now define the set

$$W_{i\to j}(t) = \left\{ h \in \Gamma_{i,j}(t) : h = \underset{k\in\Gamma_{i,j}(t)}{\arg\max}\, \mathrm{pr}(k, R_i(t)) \right\}$$

to select points of i with the maximum priority. In general $W_{i\to j}(t)$ contains more than one point. Among them we choose the ones that minimize the priority towards camera j if they were of camera j. Thus we define

$$W'_{i\to j}(t) = \left\{ h \in W_{i\to j}(t) : h = \underset{k\in W_{i\to j}(t)}{\arg\min}\, \mathrm{pr}(k, R_j(t)) \right\}.$$

Also $W'_{i\to j}(t)$ could have more than one point, in this case we choose the nearest to their future centroid $Cd(R_j(t))$ to obtain a 'fat' subregion, thus a point that belong to

$$W''_{i\to j}(t) = \left\{ h \in W'_{i\to j}(t) : h = \underset{k\in W'_{i\to j}(t)}{\arg\min}\, d(k, Cd(R_j(t))) \right\}$$

In this case all the points of $W''_{i\to j}(t)$ are equivalent for our purpose and we can exchange the first one, h^*, with respect to the total order relation of $\mathcal{Q}$. This can be done only if it's inside the physical constraints D_j.
If at time t we have two cells to exchange we can repeat this operation thus updating priority of the borderline after the exchange of the first point.
In this part of the algorithm ($|R_i(t)|-|R_j(t)\,| \geq 2$) no points can be exchanged if their priority is less or equal to 2.5. This lower bound, due to how we define the map $\mathrm{pr}(h, R)$, will lead cameras to generate straight borders avoiding the exchanging of points we priority equal or less than 2.

All of this is done to reach an equitable partition when $|R_i(t)|-|R_j(t)\,| \geq 2$. If we have $R_i(t)|-|R_j(t)| < 2$ we choose to care more on the regularization task instead of the equitable part that is almost reached. By regularization we mean that we want to decrease the average priority of both border $\Gamma_{i,j}(t)$ and $\Gamma_{j,i}(t)$. In this case we choose to distinguish the case $|R_i(t)|-|R_j(t)| = 1$ and $|R_i(t)| - |R_j(t)| = 0$. In the first one i could cover just one point of j, i.e. in this case we admit that an overlap of one point is generated (camera j doesn't release that point) and at $t+1$ those two cameras will have the same areas. In the second case two points will be exchanged (one for each camera). The selection of points to be exchanged (or covered) in both case is the same and it works like the previous case. The only difference is that we

force somehow the exchanging selecting at the beginning those points that have priority greater or equal than their priority if they will be exchanged. Obviously a camera can not add to its region points that are not inside the physical constrain. For that reason we avoid from the beginning points outside the physical constraint of the camera that have to take points.
So we start the selection from this set:

$$\bar{W}_{i\to j}(t) = \{h \in \Gamma_{i,j}(t) \cap D_j : \mathrm{pr}(h, R_i(t)) \geq \mathrm{pr}(h, R_j(t))\}$$

and for the second case also

$$\bar{W}_{j\to i}(t) = \{h \in \Gamma_{j,i}(t) : \mathrm{pr}(h, R_j(t)) \geq \mathrm{pr}(h, R_i(t))\}\,.$$

If $\bar{W}_{i\to j}(t) \neq \emptyset$ (and $\bar{W}_{j\to y}(t) \neq \emptyset$) we continue as before selecting points that have the greatest priority respect to i, than the smallest priority respect to j and finally the nearest towards j. Thus points are selected inside the final set obtained by the series of intersections:

$$W^{fin}_{i\to_j}(t) = \Big(\Big(\big((\Gamma_{i,j} \cap D_j) \cap \bar{W}_{i\to j}(t)\big) \cap W_{i\to j}(t)\Big) \cap W'_{i\to j}(t)\Big) \cap W''_{i\to j}(t)$$

and with the same criterion we obtain $W^{fin}_{j\to_i}(t)$.

In W^{fin} we can have more than one point and as before we choose the minor, we call it h^*. Now if $\mathrm{pr}(h^*, R_j(t)) > \mathrm{pr}(h^*, R_i(t))$ we can exchange the point. But if it holds $\mathrm{pr}(h^*, R_j(t)) = \mathrm{pr}(h^*, R_i(t))$ we introduce a new conditions, and we can exchange the point only if $d(h^*, Cd(R_j(t))) < d(h^*, Cd(R_i(t)))$. This condition will avoid loops where cameras keep exchanging points that have the same properties for both of them. As halting condition we pose that no points can be exchanged if their priority is less or equal than 2.5.

Before updating the subregions to $R_i(t+1)$ and $R_j(t+1)$ a check of connection is made. A disconnected result is hard to obtain but not impossible due to some specific configuration of the intersection $\Omega_{i,j}(t) \cap \Gamma_{i\to j}(t)$ that can not be avoided. This test is accomplished checking that the modified $R_i(t)$ and $R_j(t)$ (we call them $\tilde{R}_i(t)$ and $\tilde{R}_j(t)$) have all points of perimeter $\tilde{R}_i(t)$ and $\tilde{R}_j(t)$ at a distance on the graph from their centroids not infinite, thus we assure that also all the other indoor points are connected. We write

$$d_{\tilde{R}_i(t)}(P_i(t), Cd(R_i(t))) < +\infty$$
$$d_{\tilde{R}_j(t)}(P_j(t), Cd(R_j(t))) < +\infty.$$

If their not verified no update is done (thus $R_i(t+1) = R_i(t), R_j(t+1) = R_j(t)$) and cameras have to wait a future round of communication to modify their frontier.

The last step is to compute the new centroids $Cd(R_i(t+1))$ and $Cd(R_j(t+1))$. This is the computational most expensive operation because it runs on $O(n^2)$ with n number of vertex of the subregion. For that reason we choose to use euclidean distance that run O(1). Moreover we distinguished two operations: computing and updating the centroid. The first one is needed at the beginning and whenever more than two points are exchanged, for example when we resolve the overlap, and it compute the centroid considering all the points of subregions. The second one is used when just one or two points are exchanged. In this case we search the new centroids just around the old one, i.e. we test the nine vertex that surround it and we see if (13) is decreased. If so we have founded the new centroid.

4.2.2 Statement of the algorithm

The algorithm for the discretized equitable partition with a symmetric communication is stated as follows.

At each time $t \in \mathbb{Z}_{\geq 0}$, each agent $i \in 1, \ldots, N$ keeps in memory its centroid $Cd(i)$ and the environment $\mathcal{Q}$ with its subregion $R_i(t)$ and its physical constrain D_i. The collection $R(0) = R_1(0), \ldots, R_N 0$ is a covering of $\mathcal{L}$. At each time $t \in \mathbb{Z}_{\geq 0}$ agent i communicate with an adjacent agent j, where $i, j \in 1, \ldots, N$, $i \neq j$ are selected by a deterministic or stochastic process to be determined. Every agent $k \neq i, j$ sets $R_k(t+1) = R_k(t)$, while agents i perform the following tasks:

1. *% resolve overlap*
2. **if** $\Omega_{i,j}(t) \neq \emptyset$ **then**
3. $R_i(t) = \{h \in R_i(t) \cap R_j(t) : d(h, R_i(t)) \leq d(h, R_j(t))\}$
4. $R_j(t) = \{h \in R_i(t) \cap R_j(t) : d(h, R_j(t)) < d(h, R_i(t))\}$
5. compute $Cd(R_i(t)), Cd_{R_j(t)}$
6. **end**
7. *% sort cameras by area*
8. **if** $|R_i(t)| \geq |R_j(t)|$ **then**
9. $R_M = R_i(t), R_m = R_j(t), D_M = D_i, D_m = D_j$
10. **else**
11. $R_m = R_i(t), R_M = R_j(t), D_m = D_i, D_M = D_j$
12. **end**
13. *% find and exchange cells*
14. **if** $R_M - R_m \geq 2$ **then**
15. *% number n of points to be exchanged*
16. **if** $R_M - R_m > 2$ **then**
17. $n = 2$

else
  $n = 1$
end
for $k = 1 : n$
  both agents compute the set $\Gamma_{M,m}(t)$ and the sets
  $W_{M\to m}(t) = \Big\{h \in \Gamma_{M,m}(t) \cap D_m : h = \arg\max_{k\in\Gamma_{M,m}} \mathrm{pr}(k, R_M(t))\Big\}$
  $W'_{M\to m}(t) = \big\{h \in W_{M\to m}(t) : h = \arg\min_{k\in W_{M\to m}(t)} \mathrm{pr}(k, R_m(t))\big\}$
  $W''_{M\to m}(t) = \big\{h \in W'_{M\to m}(t) : d(h, Cd(R_m(t))) = \min_{k\in W'_{M\to m}(t)} d(k, Cd(R_m(t)))\big\}$
  $h^* = \min W''_{M\to m}(t)$
  if $\mathrm{pr}(h^*, R_M) \geq 2.5$ **then**
    $R_M = R_M \setminus h^*, R_m = R_m \cup h^*$
    both agents update $Cd(R_m)$, $Cd(R_M)$
  end
end
else *%* $R_M - R_m < 2$
  % R_M *gives one cell to* R_m
  both agents compute the set $\Gamma_{M,m}(t)$ and
  $\bar{W}_{M\to m}(t) = \{h \in \Gamma_{M,m}(t) \cap D_m : \mathrm{pr}(h, R_M(t)) \geq \mathrm{pr}(h, R_m(t))\}$
  $W_{M\to m}(t) = \big\{h \in \bar{W}_{m\to M}(t) : h = \arg\max_{k\in\bar{W}_{M\to m}(t)} \mathrm{pr}(k, R_M(t))\big\}$
  $W'_{M\to m}(t) = \big\{h \in W_{M\to m}(t) : h = \arg\min_{k\in W_{M\to m}(t)} \mathrm{pr}(k, R_m(t))\big\}$
  $W''_{M\to m}(t) = \big\{h \in W'_{M\to m}(t) : d(h, Cd(R_m(t))) = \min_{k\in W'_{M\to m}(t)} d(k, Cd(R_m(t)))\big\}$
  $h^* = \min W''_{M\to m}(t)$
  if $\mathrm{pr}(h^*, R_M) \geq 2.5$ **and** $\mathrm{pr}(h, R_M(t)) > \mathrm{pr}(h, R_m(t))$ **or** ($\mathrm{pr}(h, R_M(t)) = \mathrm{pr}(h, R_m(t))$ **and** $d(h, Cd(R_m(t))) < d(h, Cd(R_M(t)))$) **then**
    $R_M = R_M \setminus h^*, R_m = R_m \cup h^*$
    both agents update $Cd(R_m)$, $Cd(R_M)$
  end
  if $R_M - R_m = 0$ **then**
    % R_m *gives one cell to* R_M
    both agents compute $\Gamma_{m,M}(t)$
    $\bar{W}_{m\to M}(t) = \{h \in \Gamma_{m,M}(t) \cap D_M : \mathrm{pr}(h, R_m(t)) \geq \mathrm{pr}(h, R_M(t))\}$
    $W_{m\to M}(t) = \big\{h \in \bar{W}_{m\to M}(t) : h = \arg\max_{k\in\bar{W}_{m\to M}(t)} \mathrm{pr}(k, R_m(t))\big\}$
    $W'_{m\to M}(t) = \big\{h \in W_{m\to M}(t) : h = \arg\min_{k\in W_{m\to M}(t)} \mathrm{pr}(k, R_M(t))\big\}$
    $W''_{m\to M}(t) = \big\{h \in W'_{m\to M}(t) : d(h, Cd(R_M(t))) = \min_{k\in W'_{m\to M}(t)} d(k, Cd(R_M(t)))\big\}$
    $k^* = \min W''_{m\to M}(t)$
    if $\mathrm{pr}(k^*, R_m) \geq 2.5$ **and** $\mathrm{pr}(h, R_m(t)) > \mathrm{pr}(h, R_M(t))$ **or** ($\mathrm{pr}(h, R_m(t)) = \mathrm{pr}(h, R_M(t))$ **and** $d(h, Cd(R_M(t))) < d(h, Cd(R_m(t)))$) **then**
      $R_m = R_m \setminus k^*, R_M = R_M \cup k^*$
      both agents update $Cd(R_m)$, $Cd(R_M)$

55. **end**
56. **end**
57. **end**
58. *% checking connection of the solution*
59. **if** R_M and R_m are connected
60. *% matching correct output values*
61. **if** $|R_i(t)| \geq |R_j(t)|$ **then**
62. $R_i(t+1) = R_M, R_j(t+1) = R_m$
63. $Cd(R_i(t+1)) = Cd(R_M), Cd(R_j(t+1)) = Cd(R_m)$
64. **else**
65. $R_i(t+1) = R_m, R_j(t+1) = R_M$
66. $Cd(R_i(t+1)) = Cd(R_M), Cd(R_j(t+1)) = Cd(R_m)$
67. **end**
68. both agents update $Cd(R_i(t+1))$, $Cd(R_j(t+1))$
69. **end**

4.3 Algorithm proposed with asymmetric communication

In this section we present the algorithm developed to reach an equitable partition when the protocol of communication among cameras is asymmetric. As before we are interest not only in obtaining areas all with the same size but also with a regular shape because this aid a future patrolling strategy. As the previous case we try reach those goals together and we focus just on the regularization when two cameras has the same areas. Firstly we present the notation and the heuristic of the algorithm and after we state it.

4.3.1 Euristic and notations

Suppose that at time t camera i receives information from camera j ($i \leftarrow j$). This means that at each time of communication just one camera can update its status and in what follows that camera is i.

The first thing to do is look at the difference between areas $|R_i(t)|$ and $|R_j(t)|$. Clearly if i has more points than j it has to release some points otherwise it has to cover some points of j. If their size is the same i tries to make the border between them more regular. As before we are interested in modifying the status of just one or two points but in this case this is done also for the overlap among subregions. In fact resolving it as quickly as before with this protocol of communication leads to configurations not suitable with the regularization objective. We distinguish 3 cases.

- If $|R_i(t) - R_j(t)| \geq 2$ agent i has to release some points. To assure the covering constraint we release just points on the overlapping area (and so the physical constraint is automatically satisfied). Thus we have to start the selection inside the intersections

$$\Gamma_{i,j}^{over}(t) = \Gamma_{i,j}(t) \cap \Omega_{i,j}(t).$$

Inside $\Gamma_{i,j}^{over}(t)$ we now consider those cells that are the most irregular for i, as we want to regularize the border. So we defined

$$U_{i\to j}(t) = \left\{ h \in \Gamma_{i,j}^{over}(t) : h = \arg \max_{k \in \Gamma_{i,j}^{over}(t)} \mathrm{pr}(k, R_i(t)) \right\},$$

in general this set can have more than one point so we add this second selection that selects those cells that are more regular for j

$$U'_{i\to j}(t) = \left\{ h \in U_{i\to j}(t) : h = \arg\min_{k \in U_{i\to j}(t)} \mathrm{pr}(k, R_i j(t)) \right\},$$

also this set can have more than one point and this time we select to release those points that are more far from their centroid $Cd(R_i(t))$:

$$U''_{i\to j}(t) = \left\{ h \in U'_{i\to j}(t) : h = \arg \max_{k \in U_{i\to j}(t)} d(k, Cd(R_i(t))) \right\}$$

and finally camera i release the minor of the points inside $U''_{i\to j}(t)$ and update its centroid. If it's necessary to release two points these procedure is repeated again from the beginning to work with updated priorities.

- The second case is when $|R_j(t)| - |R_i(t)| \geq 1$ so this time camera i has to cover two (if $|R_j(t)| - |R_i(t)| > 2$) or one (if $1 \leq |R_j(t)| - |R_i(t)| \leq 2$) points of camera j. We have to find points of $R_j(t)$ adjoining to $R_i(t)$ that are not inside the overlap and we call this set $\bar{\Gamma}_{j,i}(t)$. They have to stay inside the physical constraint D_i. From that set we can take those with higher priority towards j

$$V_{j\to i}(t) = \left\{ h \in \bar{\Gamma}_{j,i}(t) \cap D_i : h = \arg\max_{k \in \bar{\Gamma}_{j,i}(t)} \mathrm{pr}(k, R_j(t)) \right\}$$

and among them those that have minor priority towards i

$$V'_{j\to i}(t) = \left\{ h \in V_{j\to i}(t) : h = \arg\min_{k \in V_{j\to i}(t)} \mathrm{pr}(k, R_i(t)) \right\}$$

and then those closer to $Cd(R_i(t))$:

$$V''_{j\to i}(t) = \left\{ h \in V'_{j\to i}(t) : h = \underset{k\in V'_{j\to i}(t)}{\arg\min}\, d(h, Cd(R_i(t))) \right\}.$$

Inside $V''_{j\to i}(t)$ we camera i covers the minor, h^*, and update its centroid. If two points are to be added to $R_i(t)$ this procedure is repeated with updated sets (i.e. posing $R_i(t) = R_i(t) \cup h^*$) in order to compute correct priorities.

There is one additional step if $|R_j(t)| - |R_i(t)| = 1$: if the two cameras has just one point of overlap that has the same priority towards both subregions and it is more near to j, camera i release that point. This is done to help the regularization in the last steps of the algorithm (when areas become almost equals and overlap almost disappear). We tolerate an increasing difference between areas just because, with this step, it actually increase of just one point (if the difference is of 2 points the algorithm won't enter in this condition). Moreover, for the same reason, i can cover the point of j only if it doesn't worsen the situation, in other words only if it has for i a lover priority or, if it's the same, it is less distant from i instead from j.

- The last case is when $|R_i(t) - R_j(t)| = 0$. The algorithm can focus on making regulars border and for that reason i tries to cover the worst point of j and releases her worst overlapping point.

  When i tries to cover the worst point of j we consider the set $\Gamma_{j,i}(t) \cap D_i$ and, if there is no overlap ($\Omega_{i,j}(t) = \emptyset$) we remove at the beginning those points that wouldn't be exchanged because $\mathrm{pr}(h, R_j(t)) > \mathrm{pr}(h, R_i(t))$. This little changing is done to force the regularization at the end, i.e. when there is no overlapping [4]. In what follow, for simplicity , we call both this set $\Gamma_{i,j}(t)$. As in the same case we find $h^* \in V''_{j\to i}(t)$ as the minor in that set. Agent i cover that point if $\mathrm{pr}(h^*, R_j(t)) > \mathrm{pr}(h^*, R_i(t))$ and if the priority is the same it checks distances covering h^* if it holds $d(h^*, Cd(R_i(t)) < d(h^*, R_j(t))$.

  When i tries to release its worst overlapping point we find $h^* \in U''_{i\to j}(t)$ and i release this point selecting checking priorities (must be $\mathrm{pr}(h^*, R_i(t)) \geq$

[4] We observed that doing this from the beginning (so when $\Omega_{i,j}(t) \neq \emptyset$) is not worthy because increasing overlap between regions may lead the algorithm to spend lots of time in situation where an equitable partition is reached, but it has a great overlap. In other words all the cameras have the same areas but their value is bigger than the optimal one reached eliminating overlap.

$\mathrm{pr}(h^*, R_j(t)))$ and if their equals checking distances (so $d(h^*, Cd(R_j(t)) < d(h^*, R_i(t)))$. Observe that this part is useful from the beginning because it regularize borders reducing overlapping. Centroid is then updated.

As with symmetric communication at the end is necessary to check if $R_i(t)$ is connected and if not the algorithm doesn't update this subregion leaving things unchanged for the next iteration $t + 1$.

4.3.2 Statement of the algorithm

The algorithm for the discretized equitable partition with asymmetric is stated as follows.

At each time $t \in \mathbb{Z}_{\geq 0}$, each agent $i \in 1, \ldots, N$ keeps in memory its centroid $Cd(i)$ and the environment $\mathcal{Q}$ with its subregion $R_i(t)$ and its physical constraint D_i. The collection $R(0) = R_1(0), \ldots, R_N 0$ is a covering of $\mathcal{L}$. At each time $t \in \mathbb{Z}_{\geq 0}$ agent i communicate with an adjacent agent j, where $i, j \in 1, \ldots, N$, $i \neq j$ are selected by a deterministic or stochastic process to be determined. Every agent $k \neq i, j$ sets $R_k(t+1) = R_k(t)$, while agents i perform the following tasks:

1. *% save conditions in t*
2. $\bar{R}_i(t) = R_i(t), \bar{R}_j(t) = R_j(t)$
3. *% First case:*
4. **if** $|R_i(t) - R_j(t)| \geq 2$ **then**
5. *% number n of points to be exchanged*
6. **if** $|R_i(t) - R_j(t)| > 2$ **then**
7. $n = 2$
8. **else** *%* $|R_i(t) - R_j(t)| = 2$
9. $n = 1$
10. **end**
11. **for** $v = 1 : n$
12. agent i compute $\Gamma_{i,j}(t)$ and the sets
13. $\Omega_{i,j}(t) = \{h \in \mathcal{Q} : h \in (R_i(t) \cap R_j(t))\}$
14. $\Gamma^{over}_{i,j}(t) = \Gamma_{i,j}(t) \cap \Omega_{i,j}$
15. $U_{i\to j}(t) = \left\{h \in \Gamma^{over}_{i,j}(t) : h = \arg\max_{k \in \Gamma^{over}_{i,j}(t)} \mathrm{pr}(k, R_i(t))\right\}$
16. $U'_{i\to j}(t) = \left\{h \in U_{i\to j}(t) : h = \arg\min_{k \in U_{i\to j}(t)} \mathrm{pr}(k, R_i j(t))\right\}$
17. $U''_{i\to j}(t) = \left\{h \in U'_{i\to j}(t) : h = \arg\max_{k \in U_{i\to j}(t)} d(k, Cd(R_i(t)))\right\}$
18. $h^* = \min U''_{i\to j}(t)$
19. **if** $\mathrm{pr}(h^*, R_m) \geq 2.5$ **then**

$R_i(t) = R_i(t) \setminus h^*$
update $Cd(R_i(t)), Cd(R_j(t))$
end
end
% Second case:
elseif $|R_i(t) - R_j(t)| \geq 1$ **then**
% number n of points to be exchanged
if $|R_i(t) - R_j(t)| > 2$ **then**
$n = 2$
else *%* $1 \leq |R_i(t) - R_j(t)| \leq 2$
$n = 1$
end
compute $\Omega_{i,j}(t) = \{h \in \mathcal{Q} : h \in (R_i(t) \cap R_j(t))\}$
if $|R_i(t) - R_j(t)| = 1$ **and** $\Omega_{i,j}(t) \equiv h$ **then**
if $\mathrm{pr}(h, R_i(t)) = \mathrm{pr}(h, R_j(t))$ **and** $d(h, R_j(t)) < d(h, R_i(t))$ **and** $\mathrm{pr}(h, R_m) \geq 2.5$ **then**
$R_i(t) = R_i(t) \setminus h$
update $Cd(R_i(t))$
end
end
for $v = 1 : n$
agent i compute $\Gamma_{j,i}(t)$ and the sets
$\bar{\Gamma}_{j,i}(t) = \{\Gamma_{j,i}(t) \cap D_i\} \setminus \{h \in \Gamma_{j,i}(t) : \mathrm{pr}(h, R_j(t)) > \mathrm{pr}(h, R_i(t))\}$
$V_{j\to i}(t) = \left\{h \in \bar{\Gamma}_{j,i}(t) : h = \arg\max_{k \in \bar{\Gamma}_{j,i}(t)} \mathrm{pr}(k, R_j(t))\right\}$
$V'_{j\to i}(t) = \left\{h \in V_{j\to i}(t) : h = \arg\min_{k \in V_{j\to i}(t)} \mathrm{pr}(k, R_i(t))\right\}$
$V''_{j\to i}(t) = \left\{h \in V'_{j\to i}(t) : h = \arg\min_{k \in V'_{j\to i}(t)} d(h, Cd(R_i(t)))\right\}$
$h^* = \min V''_{j\to i}(t)$
if $|R_i(t) - R_j(t)| \neq 1$ **then**
$R_i(t) = R_i(t) \cup h^*$
update $Cd(R_i(t))$
elseif ($\mathrm{pr}(h^*, R_j(t)) < \mathrm{pr}(h^*, R_j(t))$ **or** ($\mathrm{pr}(h^*, R_j(t)) = \mathrm{pr}(h^*, R_j(t))$ **and** $d(h^*, R_i(t)) < d(h^*, R_j(t))$)) **and** $\mathrm{pr}(h^*, R_m) \geq 2.5$ **then**
$R_i(t) = R_i(t) \cup h^*$
update $Cd(R_i(t))$
end
end
% Third case:
else *%* $|R_i(t) - R_j(t)| = 0$
% agent i cover

57. **if** $\Omega_{i,j}(t) = \emptyset$ **then**
58. compute $\Gamma_{j,i}(t) = \{\Gamma_{j,i}(t) \cap D_i\} \backslash \{h \in \Gamma_{j,i}(t) : \mathrm{pr}(h, R_j(t)) > \mathrm{pr}(h, R_i(t))\}$
59. **end**
60. $V_{j\to i}(t) = \left\{h \in \Gamma_{j,i}(t) : h = \arg\max_{k \in \bar{\Gamma}_{j,i}(t)} \mathrm{pr}(k, R_j(t))\right\}$
61. $V'_{j\to i}(t) = \left\{h \in V_{j\to i}(t) : h = \arg\min_{k \in V_{j\to i}(t)} \mathrm{pr}(k, R_i(t))\right\}$
62. $V''_{j\to i}(t) = \left\{h \in V'_{j\to i}(t) : h = \arg\min_{k \in V'_{j\to i}(t)} d(h, Cd(R_i(t)))\right\}$
63. $h^* = \min V''_{j\to i}(t)$
64. **if** ($\mathrm{pr}(h^*, R_j(t)) > \mathrm{pr}(h^*, R_i(t))$ **or** ($\mathrm{pr}(h^*, R_j(t)) = \mathrm{pr}(h^*, R_i(t))$ **and** $d(h^*, Cd(R_i(t)) < d(h^*, R_j(t))$)) **and** $\mathrm{pr}(h^*, R_m) \geq 2.5$ **then**
65. $R_i(t) = R_i(t) \cup h^*$
66. update $Cd(R_i(t))$
67. **end**
68. *% agent i cover*
69. compute the sets $\Gamma^{over}_{i,j}(t) = \Gamma_{i,j}(t) \cap \Omega_{i,j}(t)$
70. $U_{i\to j}(t) = \left\{h \in \Gamma^{over}_{i,j}(t) : h = \arg\max_{k \in \Gamma^{over}_{i,j}(t)} \mathrm{pr}(k, R_i(t))\right\}$
71. $U'_{i\to j}(t) = \left\{h \in U_{i\to j}(t) : h = \arg\min_{k \in U_{i\to j}(t)} \mathrm{pr}(k, R_i j(t))\right\}$
72. $U''_{i\to j}(t) = \left\{h \in U'_{i\to j}(t) : h = \arg\max_{k \in U_{i\to j}(t)} d(k, Cd(R_i(t)))\right\}$
73. $h^* = \min U''_{i\to j}(t)$
74. **if** ($\mathrm{pr}(h^*, R_i(t)) > \mathrm{pr}(h^*, R_j(t))$ **or** ($\mathrm{pr}(h^*, R_i(t)) = \mathrm{pr}(h^*, R_j(t))$ **and** $d(h^*, Cd(R_j(t)) < d(h^*, R_i(t))$)) **and** $\mathrm{pr}(h^*, R_m) \geq 2.5$ **then**
75. $R_i(t) = R_i(t) \cup h^*$
76. update $Cd(R_i(t))$
77. **end**
78. **end**
79. *% checking connection of the solution and updating*
80. **if** $R_i(t)$ and R_j are connected **then**
81. $R_i(t+1) = R_i(t), R_j(t+1) = R_j(t)$
82. **else**
83. $R_i(t+1) = \bar{R}_i(t)), R_j(t+1) = \bar{R}_j(t)$
84. **end**

4.4 Computational complexity

In this section, we describe the computational requirements of algorithms developed for a symmetric and asymmetric communication. Due to their similar approach to the problem computational issues are the same for both of them and are in what follows discussed. The assumption is that the

environment is discretized using an occupancy grid map, the fact that all edge weights are the same for these graphs results in a significant computational savings.

4.4.1 One-to-all distances

Computing distances from one vertex to all other vertices in a subgraph of G (i.e. one-to-all distances) is the core computation of the algorithm. For graphs used in occupancy grids each cell can have a maximum of 4 edges, so all computational bounds can be stated in terms of $|Q|$. In addition, the uniform weight of edges means that computing distances in the graphs is equivalent to counting hops. We can therefore use a Breadth First Search (BFS) approach to compute one-to-all distances on the fly in $O(|Q|)$ in both time and memory. However if is necessary to work with different edge weights then Dijkstra's algorithm must be used (it requires $O(|Q|\log(|Q|))$ in time and $O(|Q|)$ in memory). In literature there is also a solution for small graphs that can be The Johnson's algorithm pre-compute all pairwise distances between vertices in $(O(|Q|^2\log(|Q|)))$ and then use a constant-time lookup online. However, the memory requirements is $O(|Q^2|)$ and for that reason it's useless for large environments.

4.4.2 Exchanging territory

There are three stages to the computation of which territory cells to exchange. In what follows we illustrate those steps for a symmetric protocol where both agents have to perform those computation and transmission. With an asymmetric protocol the transmission is just on one camera and the computation just to the other.

So with a symmetric protocol agents i and j must exchange their current subsets p_i and p_j, requiring a transmission of $O(|p_i|+|p_j|)$ bits of information. Secondly, for update their subset they must find the border between them and find the best cells to exchange. These computation require $O(|p_i|+|p_j|)$ in both time and memory. Finally, each agent must check that the result are connected sub-graphs and compute centroid $Cd_i(t)$ and $Cd_j(t)$ and this will be discussed below.

With an asymmetric protocol agents i must exchange its current subset p_i and it requires a transmission of $O(|p_i|)$ bits of information. After that j have to find the cells to cover (or to release) and it requires $O(|p_i|+|p_j|)$ in both time and memory. Then a check of connection and the compute of new centroid $Cd_j(t)$ have to be made.

4.4.3 Check of connection and centroid computation

These two steps are the most computationally demanding. Check connection means checking that exists a path from every node to the centroid. In order to reduce the computational requirements of this step the connection is checked just for vertexes on the border P_i and P_j. This is sufficient to guarantee the connection of the areas. Using Dijkstra's algorithm we obtain a complexity of $O(|P_i + P_j|)\log(|P_i + P_j|)$ in time and $O(|P_i + P_j|)$ in memory with symmetric communication. Clearly for the asymmetric version this check requires $O(|P_j|\log(|P_j|))$ in time and $O(|P_j|$ in memory .

The problem of computing centroid is the presence of local minima in minimizing the sum of distances to all other vertices. A first approach is making an exhaustive search. Find a vertex in p_i with the minimum cost requires computing the one-to-all distances for each vertex. This approach finds the true centroid but requires $O(|p_i|^2)$ time. A second approach used is the gradient descent. If the cells exchanged are one or two the new centroid must be in one of the eight cells that surround the previous centroid, so the one-to-all distances can be computed just around the previous centroid. In both algorithms when agents perform an exchange of just one or two points this second approach is used. In all the other cases an exhaustive search is performed to assure the choice of the right centroid. Observe that with the asymmetric version the second approach is almost always enough to find centroids. In fact just initialization and the management of a faulting camera require to use an exhaustive search. Whereas with symmetric communication it depends on the size of overlap (if present). So the choice of exchanging just few vertexes leads to the final solution in more iterations but with a great computational saving on the core of computation of the algorithm.

5 Simulation results for 2D case with symmetric communication

In this section we report the results obtained running the discretized equitable partitioning in a two dimensional environment with symmetric communication explained in Section 4.2.

In order to test the algorithm we choose a square area discretized with 15×15 cells that has to be patrolled by 9 cameras. We choose this numbers in order to make possible an optimal partition where every camera has to patrol exactly 25 cells. Note that talking about cameras a rough discretization (i.e. with a small number of cell respect to the size of the surface) can be chosen and it will speed up the ending of the transitory of algorithm. In a real application the size of the cell is a parameter of the design and it can vary according to specifics purposes.

As initial conditions every subregion assigned to a camera overlap its neighobors, thus assuring that the covering constraints is satisfied. It make sense in a real application where, for example, we have to install a network of cameras and we don't want to spend too much time in assigning the initial subregions to all of the cameras. With this algorithm we can quite roughly assign a region to every camera (the only constrain is that there haven't to be unpatrolled spaces) because the algorithm, as we will see, it's robust enough to initial conditions.

5.1 A balanced and not constricted configuration

In the next figures we can see an example of how the algorithm works. In what follows the area to patrol is the biggest square and to each camera is assigned a subregion represented by cells (the smallest square) of different colours. The overlap is simply represented by cells nested one in the other and black triangles are the centroids of subregions. A black arrow is used to explicit the pair of communicating cameras in a round of communication. In Figure 14a we can see that the starting point is represented by the entire area partitioned and big parts of it have to be patrolled by more than one camera. We consider this a quite balanced initialization, almost all of the overlaps are not so big and centroids are posed more or less where they will be in the optimal result. This make sense because in practice, working with cameras, we have physical constraints that limits the competence of the cameras to a limited subregion.
In Figure 14b-14f we observe the behaviour of the algorithm. We can say that in a first phase the more relevant action is the one of resolving overlapping

between adjacent subregions. In this step the size of areas could have variations quite significant (it depends on how the system has been initialized) and we have the greatest variation of the performance index $\varepsilon(t)$.
Until the ending of this phase, i.e. when all initial overlaps are disappeared, its possible that the index $\varepsilon(t)$ increase significantly (see Figure 14a) and it happens because overlap is splitted checking the distances and this means that also the biggest area can increment its value if needed [5].
In the second part cameras make the index to converge almost to zero. We say almost because neighbors cameras with areas of the same size can think on regularize their borderline maybe incrementing a little bit the index as we've seen before.
The last part is when the index is really close to zero. This can last quite a lot because it depends on the communication that is random, there are just few cameras that need to communicate in order to fix the last changing on border but we have to wait that the random process that rules the communication selects them. If it selects other cameras the round stay unused because no changing are needed between them.

Figure 15a shows how change the index

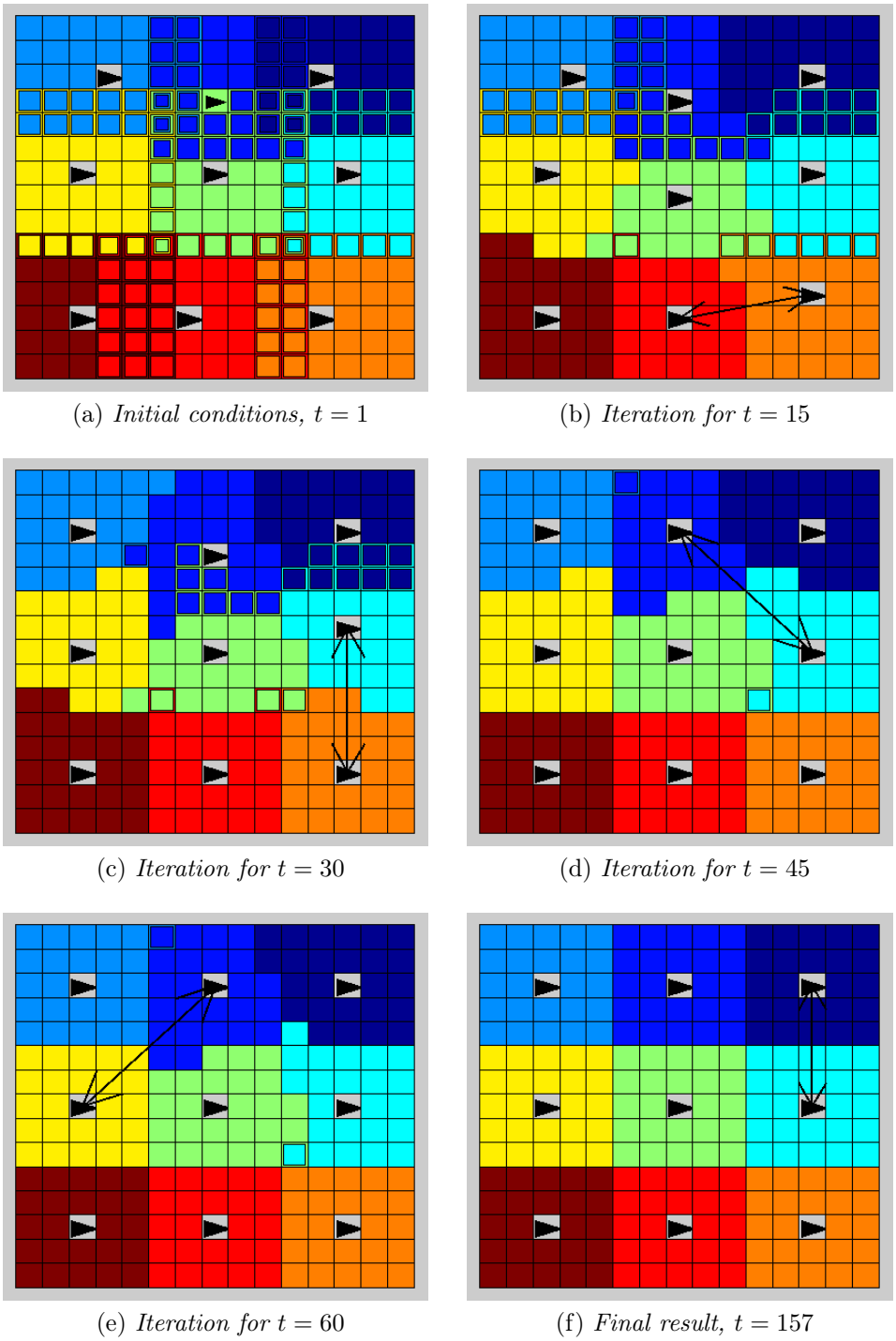
$$\psi_i(t) = \frac{1}{|\Lambda(R_i(t))|} \sum_{h \in \Lambda(R_i(t))} d_{R_i(t)}(h, Cd(R_i)) \, , i \in \{1, \ldots, N\} \qquad (15)$$

where $\Lambda(R_i(t))$ is the set of all the points that lay on the perimeter. It's clear that the less is the mean of the distance from each point of the perimeter to its centroid the fatter is the shape. We can see that they all decrease for every camera until they all reach the same value.

5.2 Unbalanced initial configuration

In the previous example we saw the behaviour starting with a quite good initialization that can be justified by the presence of physical constraints that limits the areas of the cameras. In a case like this a natural initialization is to assign to all cameras their physical constraints. Different is the case where agents are not fixed cameras but mobile agents, for example mobile robots

[5] We chose to solve overlap present in this context looking just on distances form centroids. This is equivalent to say that this choice is made favouring the part of regularization instead of the one of equitable. This is supposed to avoid some cases where a smaller area win all the overlap but makes the bigger area to assume a shape difficult to regularize and sometimes easily to be disconnected in the future. Gaining point on the distance from centroid is more suitable to give 'fat' areas easier to work with. In other words we chose to work with a more regular but less balanced 'initial' condition instead of starting with bad shapes and more balanced areas.

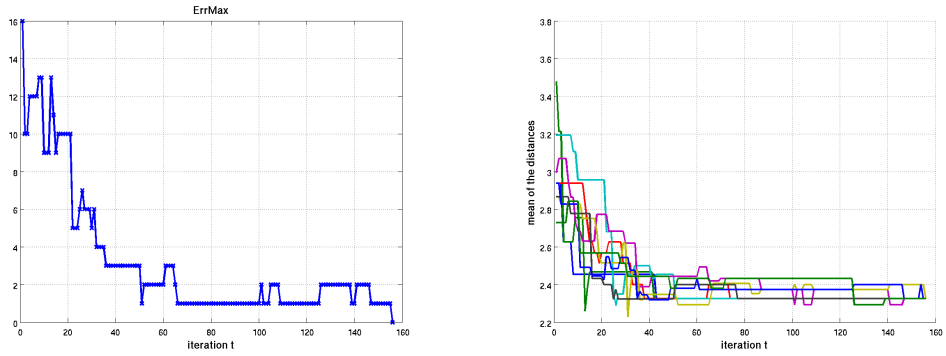


(a) *Initial conditions,* $t = 1$ (b) *Iteration for* $t = 15$

(c) *Iteration for* $t = 30$ (d) *Iteration for* $t = 45$

(e) *Iteration for* $t = 60$ (f) *Final result,* $t = 157$

Figure 14: Some passages of the algorithm to obtain an equitable partition of a discretized environment with a symmetric communication among 9 cameras.



(a) *Plot of the maximum difference* $\varepsilon(t)$ *among areas size*

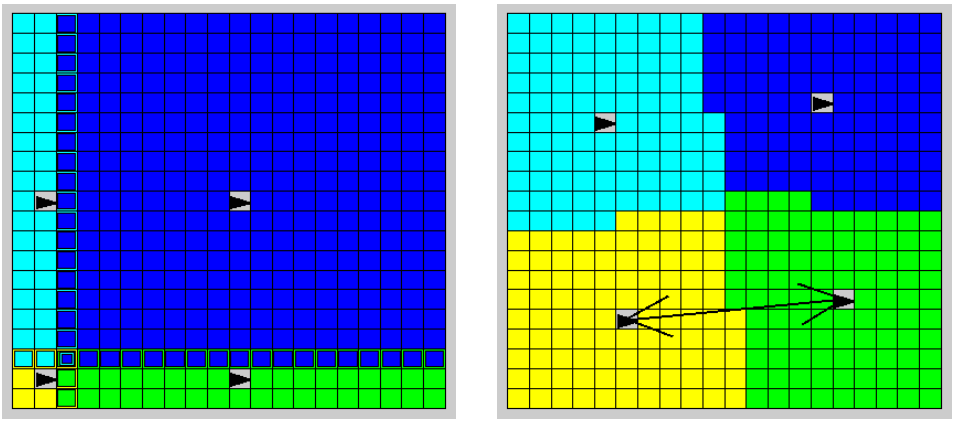
(b) *Plot of the mean of distance* $\psi_i(t)$ *between perimeter and centroids for all cameras*

Figure 15

that can go everywhere on the surface. In this case the centroid represent the position of the robot and obtaining a 'fat' subregion is fundamental to minimize the time needed by robot to reach the farthest point in its subregion. In this case make more sense to consider an initial condition very unbalanced that must be regularized also by significant movements of agents. We choose as instance an environment (see Figure 16a) with an edge of 20 cells and 4 agents inside. The final result is shown in Figure 16b, the optimal result as we saw it before it's not perfectly reached but we get a still good approximation of it. From this point the algorithm cannot move on further and it remain stucked in this sub-optimal configuration. Comparing Figure 17a and 17b we can see that an equitable partition is reached a little bit after $t = 160$ but at this time indexes $\psi_i(t)$ are quite far from being the same. For that reason perimeters continue to be modified by agents until they obtain all $\psi_i(t) \simeq 5.2$. There is a cell that is cyclic exchanged among agents that try to reach the optimal solution $\psi_i(t) = \psi_j(t), i \neq j, t \geq \tau$ and $\varepsilon(t) = 0, t \geq \tau$ for a given τ. This is no more possible because we tolerate that a camera can exchange no more than one point 'against' the equitable and that's not enough to change the global situation. The amplitude of oscillations are limited and kept very small as we see in Figure 17c and 17d.

5.3 Possible disconnected configurations

In the next example we see how it is possible that an area became disconnected and so the importance of making a check on disconnection before



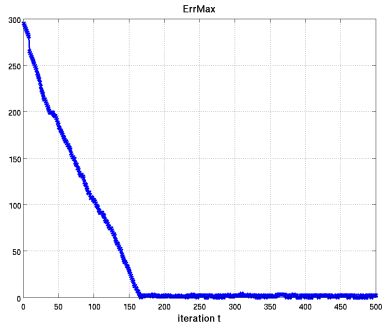
(a) *Initial conditions,* $t = 1$ (b) *Final result at* $t = 500$

Figure 16: Result of the algorithm starting with a strongly unbalanced initial condition.

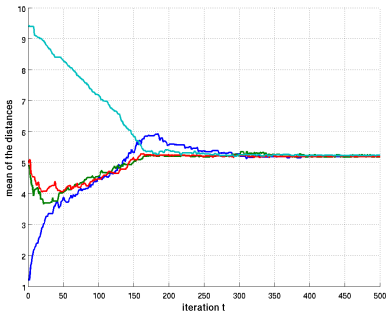
updating subregions. The starting point is shown in the Figure 18a. The green camera communicating first with the yellow and after with the brown one gains two points. This is the time reported, now if the yellow releases the overlap point to the green a disconnection is generated and because of the disconnected part, it will keep on communicating with the red camera also if their centroids are not 'neighbour'. In Figure 18b the situation goes on the yellow can catch point from the red camera but can not release the point that has in common with the green one. In order to unlock that point we have to wait that or the green one goes back or the yellow goes back and leave all that points to the red one. To reach a more regular situation the second option is exactly what happens. In Figure 18c the red starts advance towards the yellow catching a point previously of the brown one. The situation evolves in Figure 18d and 18e where red and brown fix themselves and then the green fix itself avoiding its own disconnection. Finally in Figure 18f the optimal result is obtained.

5.4 Small physical constraints

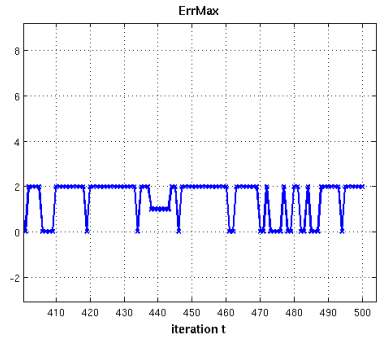
We see now what happen when there are physical constraints that became actives and in particular some camera cannot fully provide its contribute to reach an equitable partition. We take the square area with edges discretized to 15 cells. We've already seen that each of the nine camera should take its sub-area to 25 points, but we suppose now that two cameras have a maximum reachable surface of 16 points. In Figure 19a we initialize every



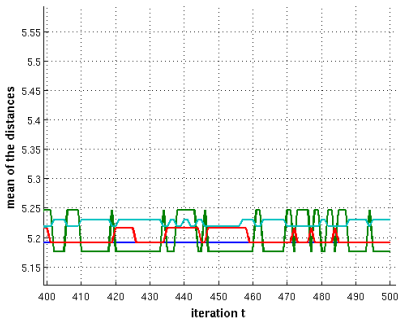
(a) *Plot of the maximum difference* $\varepsilon(t)$ *among areas size*



(b) *Plot of the mean of distance* $\psi_i(t)$ *between perimeter and centroids for all cameras*

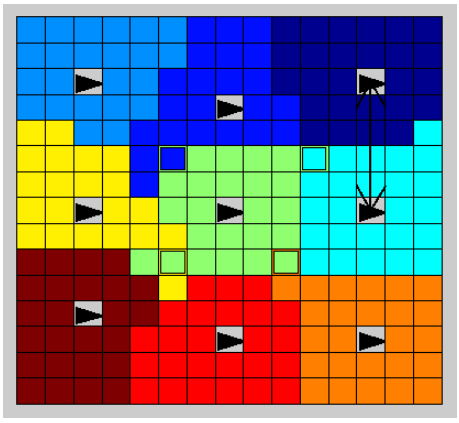


(c) *Zoom of final oscillations of* $\varepsilon(t)$

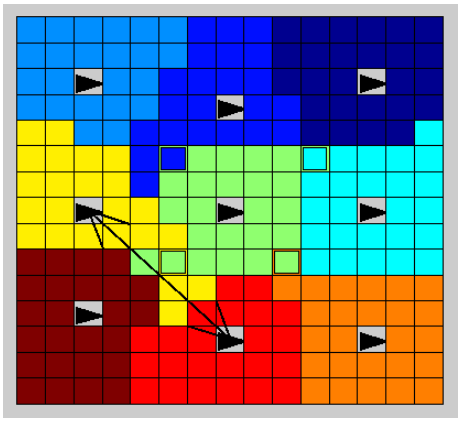


(d) *Zoom of final oscillations of* $\psi_i(t)$

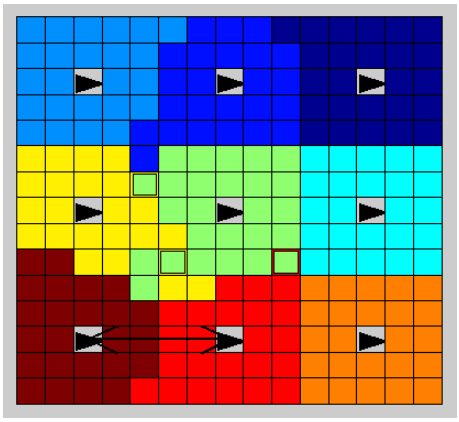
Figure 17: Maximum error $\varepsilon(t)$ and index $\psi_i(t)$ (and their zooms) for an environment patrolled by 4 agents with an unbalanced initialization.



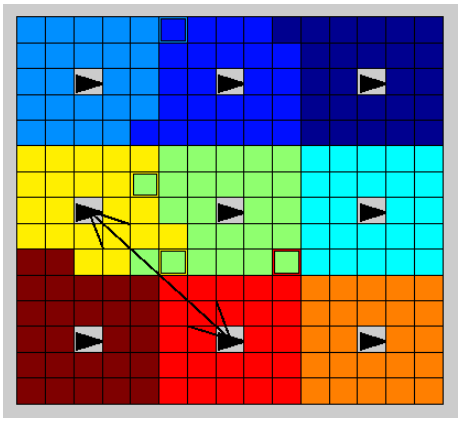
(a) *Beginning of a potential disconnection,* $t = 55$



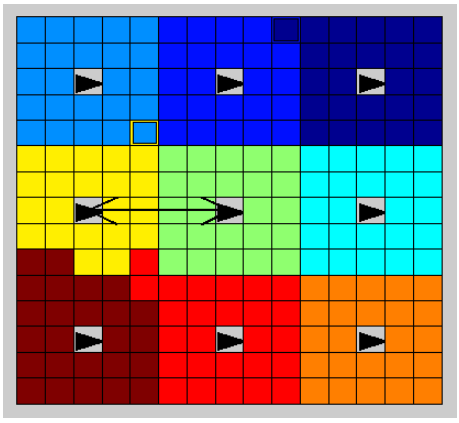
(b) *The advancing of yellow camera, overlap stay locked,* $t = 56$



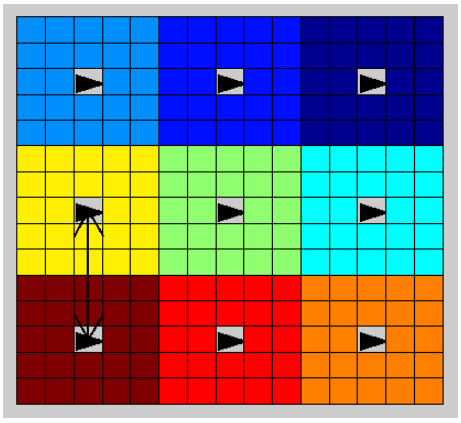
(c) *Red and brown fixing their border,* $t = 92$



(d) *Yellow moving backwards,* $t = 105$

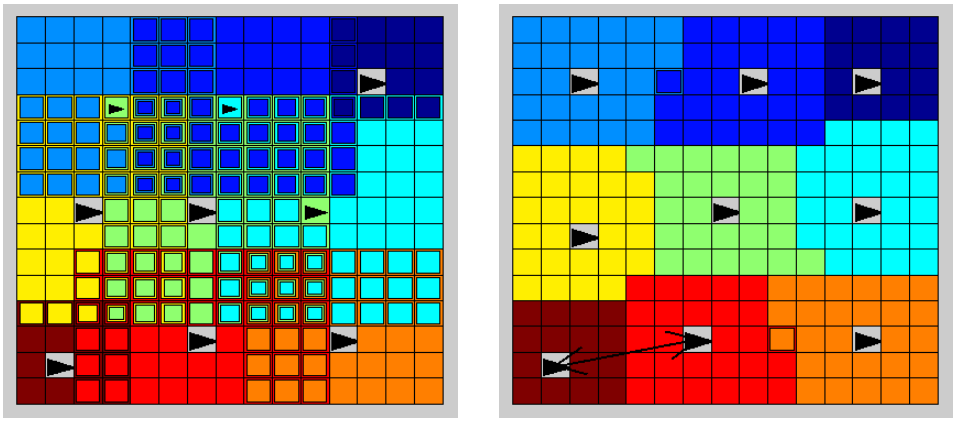


(e) *Green solving its border,* $t = 114$



(f) *Final result,* $t = 159$

Figure 18: Some passages of the algorithm when a possible disconnection situation appears.



(a) *Initialization of cameras with their physical constraints*

(b) *Final result*

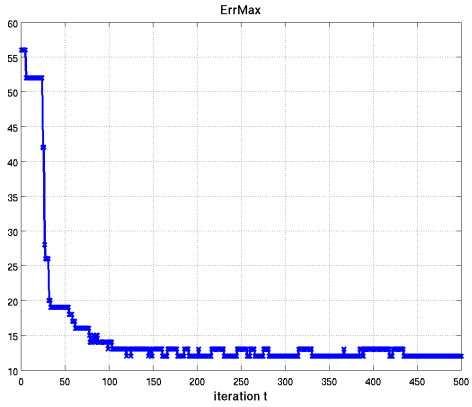
Figure 19: An example where physical constraints don't contain the optimal solution (see brown and dark-blue areas).

camera with its physical constraint and the more constrained cameras are the first and the last one (respectively in brown and dark blue). In a case like this its clear that the two constrained cameras will be stay fixed to their maximum spread and the other ones have to take more cells to cover the whole environment (accordingly to their possibilities). Moreover they reach a partition that divide equally (if possible) the part of the environment outside fixed cameras. Note that final result can have some points of overlap left or some irregularities. This is mainly because the regularization action is performed in a place where it's not possible to have a perfect equal partition. In fact the number of cells that lays outside 'fixed' cameras it's not a multiple of the number of cameras that can adjust their areas. We remark that, that difference consist only in a few cells (usually less than two) as shown in Figure 20b. We report also the index $\varepsilon(t)$ in Figure 20a that of course cannot go to zero but reaches, oscillating a little bit during regularization, the best (i.e. 12 cells) that is possible to do [6].

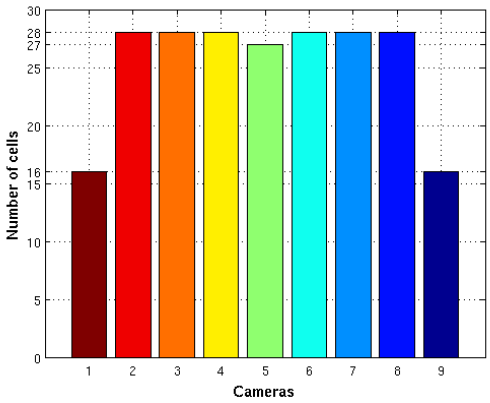
5.5 Fault of a camera

In the next example we will deal with the problem of the fault of one camera. This could be due or to a malfunctioning of the agent but also

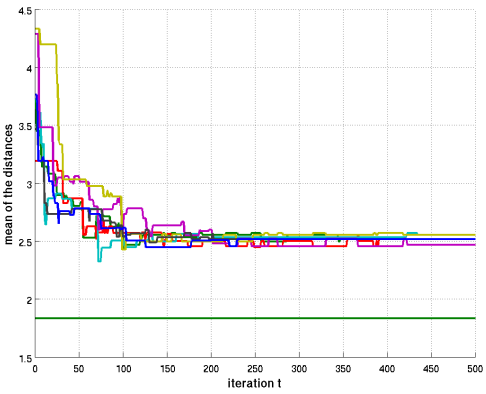
[6]The whole area has 255 cells. Two cameras are fixed with 16 points each due to constraints. So we have $225 - 16\dot{2} = 193$ and divided by the 7 'free' agents leads to 27.57 per agent. So the minimum difference that we can obtain is $27.57 - 16 = 11.57$!



(a) *Evolving of the index $\varepsilon(t)$, it settles to the final value of* 12



(b) *Size of the area assigned to each camera in the final configuration*



(c) *Index $\psi_i(t)$ for $i = 1, \ldots, N$ the two more constrained cameras stay fixed, the others try to settle themselves to the same shape*;

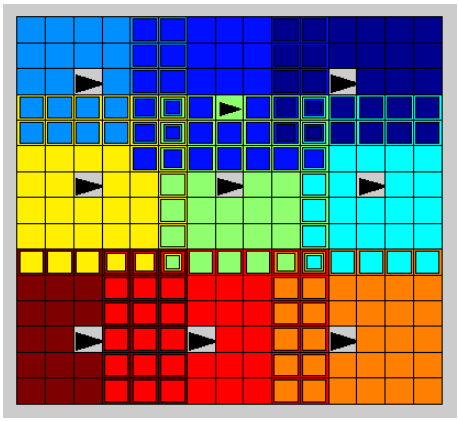
Figure 20: An example where physical constraints don't contain the optimal solution (see brown and dark-blue areas).

to a situation where the agent is busy in following an object (or in general providing a service) and cannot continue the task of patrolling its area. In a case like this it's clear that all neighbours agents have to make up for the one occupied. The idea is of expanding borders up to their constraints for all cameras in order to recompute an optimal partition that can compensate for the lack of one camera. Note that in this part we don't deal with the problem of how actually the faulting camera inform all the others of its absence. It's possible to implement this function asking that the camera that is going to miss have to advise at least on neighbour and it will provide this information in a distributed way throughout all the network[7] or in some other ways. In Figure 21 we reported an example where the eighth camera stop working (or becomes busy) in $t = 101$. At that time cameras were trying to reach an optimal configuration (see Figure 21b to see configuration at $t = 100$), but because of the fault in $t = 101$ they re-initialize themselves up to their physical constraints to affrontare the new situation. We know that each camera should patrol 25 cells in order to obtain $\varepsilon(t) = 0$ but now there are 9 cells that cannot reached so the new best configuration requires that cameras have 27 cells in their subregions. Being a camera to 0 we hope that $\varepsilon(t) = 27$ during the interval where a camera is missing. In this case a sub-optimal configuration is reached with $\varepsilon(t) = 28$, so there is almost one camera that has a point more than necessary. The explanation can easily gained looking at Figure 22b, in fact the difference of one point is because they haven't reached a minimum $\psi_i(t)$ and they keep working on it (spending no more than one point). In 22b we can also see the effects of re-initialization at $t = 101$ and the re-appearing of the faulting camera at $t = 401$. That partition is shown in Figure 21d and in the last two Figure 21e and 21f we can see the optimum result reached at the end.

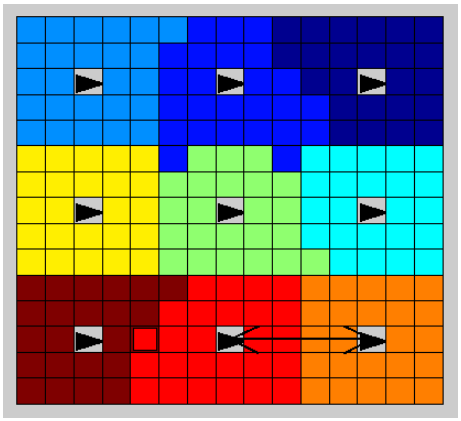
5.6 Statistical results

Finally we present results obtained testing the control law in an environment similar to the one used before (a discretized square with an edge of 15 cells) but with a random initialization of agents that can vary from a smooth to a very unbalanced initialization. In order to have an idea of the general behaviour of the algorithm we run a hundred of simulations reaching the optimal configuration 67 times (in, on average, 538 iterations). For the 33 cases where the optimum is not reached we analysed the index ψ to see

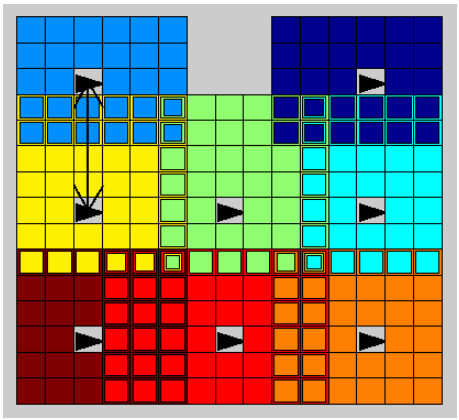
[7] This can be fulfilled, for example, if every camera keeps in memory a vector of N elements where it stores information about their status and if there are updates they must be sent. Areas are modified when all cameras have that vector updated - so also this information must be send -.



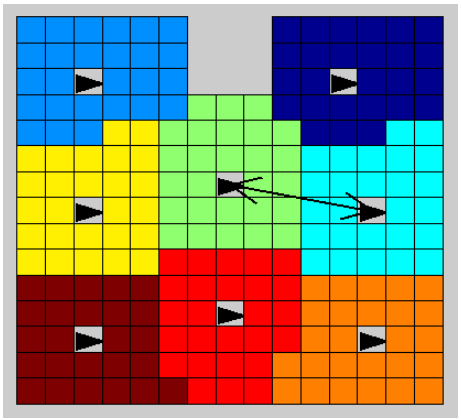
(a) *Initial condition,* $t = 1$



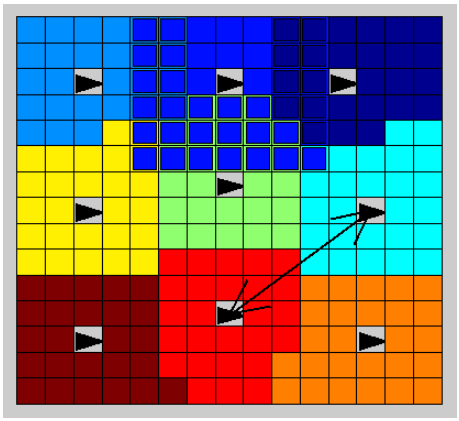
(b) *Configuration just before a camera fault,* $t = 100$



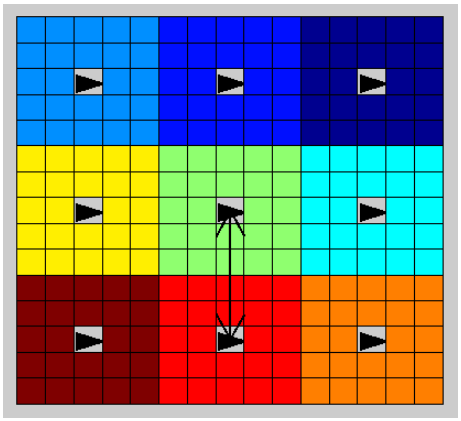
(c) *Reinitialization cause of the fault of the middle-blue camera,* $t = 101$



(d) *Final result with a camera missing,* $t = 400$

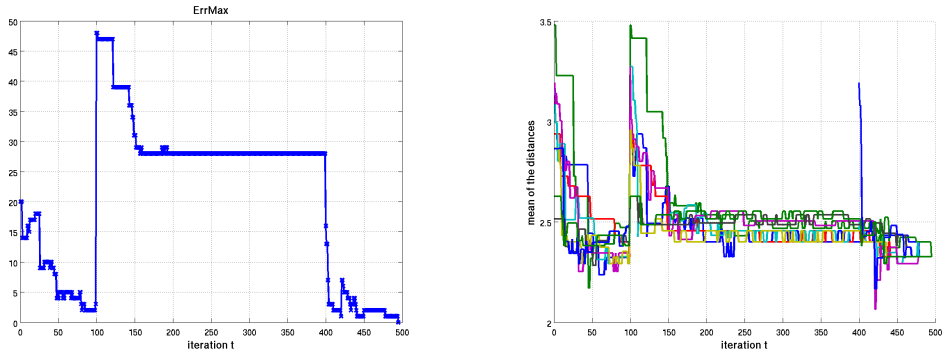


(e) *The middle-blue camera reappears spreading its physical constrain ,* $t = 401$



(f) *Final result with all cameras,* $t = 496$

Figure 21: Main passages of the algorithm when a fault of a camera appears.



(a) *Index $\varepsilon(t)$, evident are points where a camera fault $t = 100$ and when it begins to work again $t = 400$*

(b) *Progress of indexes $\psi_i(t)$*

Figure 22: An example where physical constraints don't contain the optimal solution (see brown and dark-blue areas).

the performance. As seen before it measures the mean of the distance of every point of the perimeter from the centroid. In the optimal configuration we have $\psi_{opt} = 2.325$ for each camera. To see how many solutions that don't reach the optimum are far from that result we measure the quantity $\eta(T) = \max_i \psi_i(T) - \psi_{opt}$, where T is the final instant. Considering all the 33 cases we have:

$$\mathbb{E}[\eta] = 0.183$$
$$\sigma^2(\eta) = 0.019$$

so we can affirm that on average if the optimum is not reached a sub-optimal it is (at less than 7%). There are still some exceptions where the algorithm converge to an equitable partition but assigning to some cameras sub-regions with a bad shape. Note that in this test all cameras are considered without physical constraints to see the evolution of their area's shapes. In a scenario with physical constraints areas are limited by that boundaries and those limits can help the reaching of the optimal solution keeping areas not too far from it. In Figure 23 where are reported η for all cases.

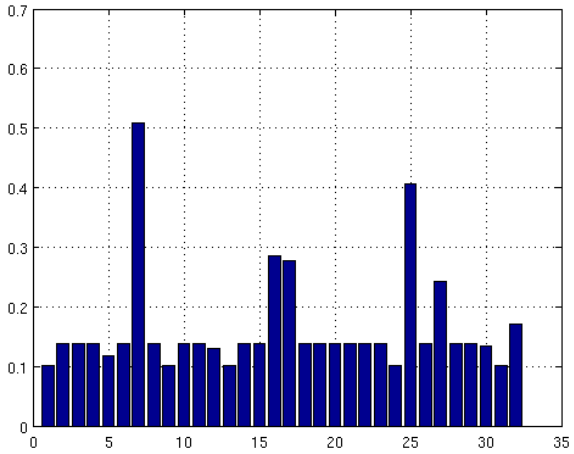


Figure 23: Difference η between the worst ψ among all cameras and ψ_{opt}

6 Simulation results for 2D case with asymmetric communication

In this section we present results obtained running the algorithm developed for equitable in a discretized environment using an asymmetric protocol of communication among agents that perform the partitioning.

The environment used its the same of the previous case with symmetric communication, i.e. a square area discretized in 225 cells that have to be partitioned (and then of course patrolled) by 9 cameras.

6.1 Balanced and not constricted initialization

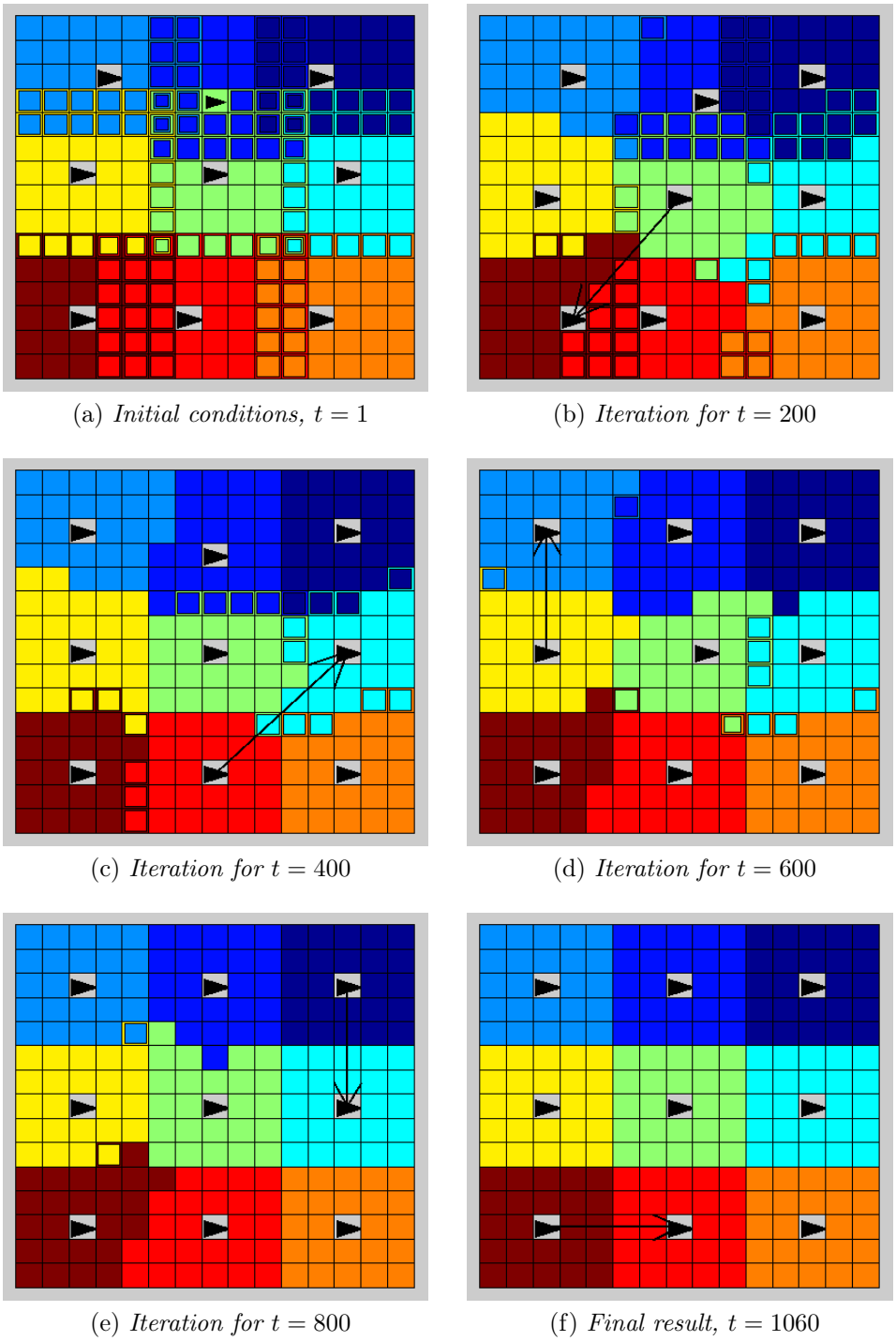
We start showing a general behaviour when cameras have a quite balanced initialization and there aren't physical constraints. In Figure 24 there are reported some moments of the simulation. We see that differently from the case with symmetric communication the management of overlapping areas is here slower (in particular at the beginning) and in order to keep every part patrolled it's necessary that each camera modifies its subregion catching and releasing overlapping cells with its neighbours. Note that now the arrow used as to show the communication has just one head pointing to the camera that is receiving information. In Figure 25a is shown the progress of the difference between the biggest and smallest subregion and in Figure 25b the progress of the index ψ that provides us information about subregions shapes. Observe that $\epsilon(t)$ reaches the value zero at the end but also in a previous instant (when all subregions have the same size but there is overlap among them) and that all ψ_i converge to the same value.

6.2 Unbalanced unconstrained initialization

In the previous example we supposed a quite balanced initialization, the algorithm works well also with an unbalanced initializations. Clearly observations reported in Section 5 for the symmetric case are still valid. The initialization and the final result is reported in Figure 26a and 26b and the progress of indexes are in Figure 27a and 27b. In this simulation the partition is sub-optimal in terms of shapes with $\psi_i(T)$ that are almost the same ($\psi_3(T) = \psi_4(T) = 5.203$ and $\psi_1(T) = \psi_2(T) = 5.207$).

6.3 Possible disconnected configurations

In what follows we see with an example the importance of checking the connection of the result in each iteration of the algorithm. In this case

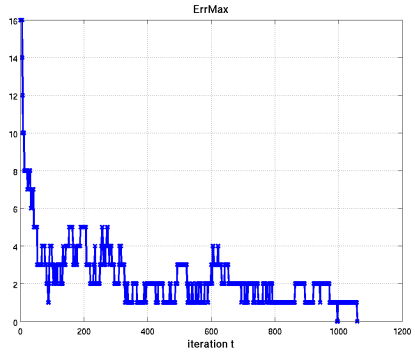


(a) *Initial conditions,* $t = 1$ (b) *Iteration for* $t = 200$

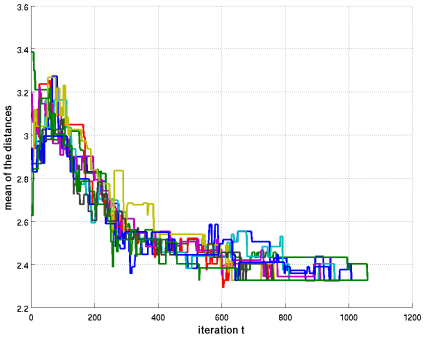
(c) *Iteration for* $t = 400$ (d) *Iteration for* $t = 600$

(e) *Iteration for* $t = 800$ (f) *Final result,* $t = 1060$

Figure 24: Some passages of the algorithm to obtain an equitable partition of a discretized environment with a symmetric communication among 9 cameras.

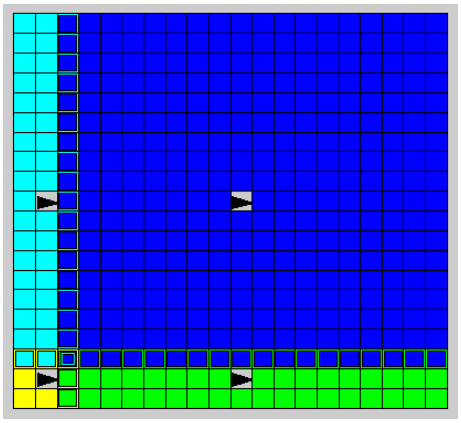


(a) *Plot of the maximum difference* $\varepsilon(t)$ *among areas size*

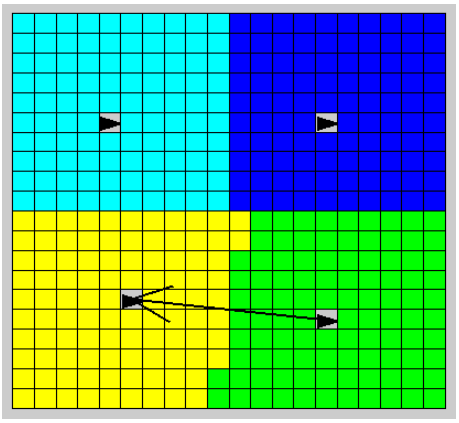


(b) *Plot of the mean of distance* $\psi_i(t)$ *between perimeter and centroids for all cameras*

Figure 25: Progress of indexes for a balanced and not constricted initial condition.

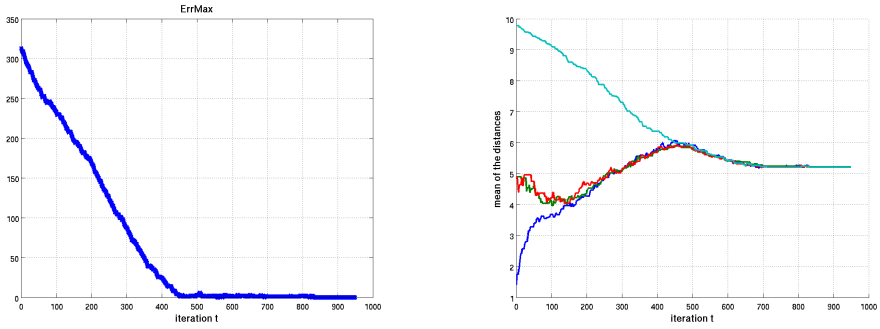


(a) *Initial conditions,* $t = 1$



(b) *Final result at* $t = 950$

Figure 26: Result of the algorithm starting with a strongly unbalanced initial condition.



(a) *Plot of the maximum difference $\varepsilon(t)$ among areas size*

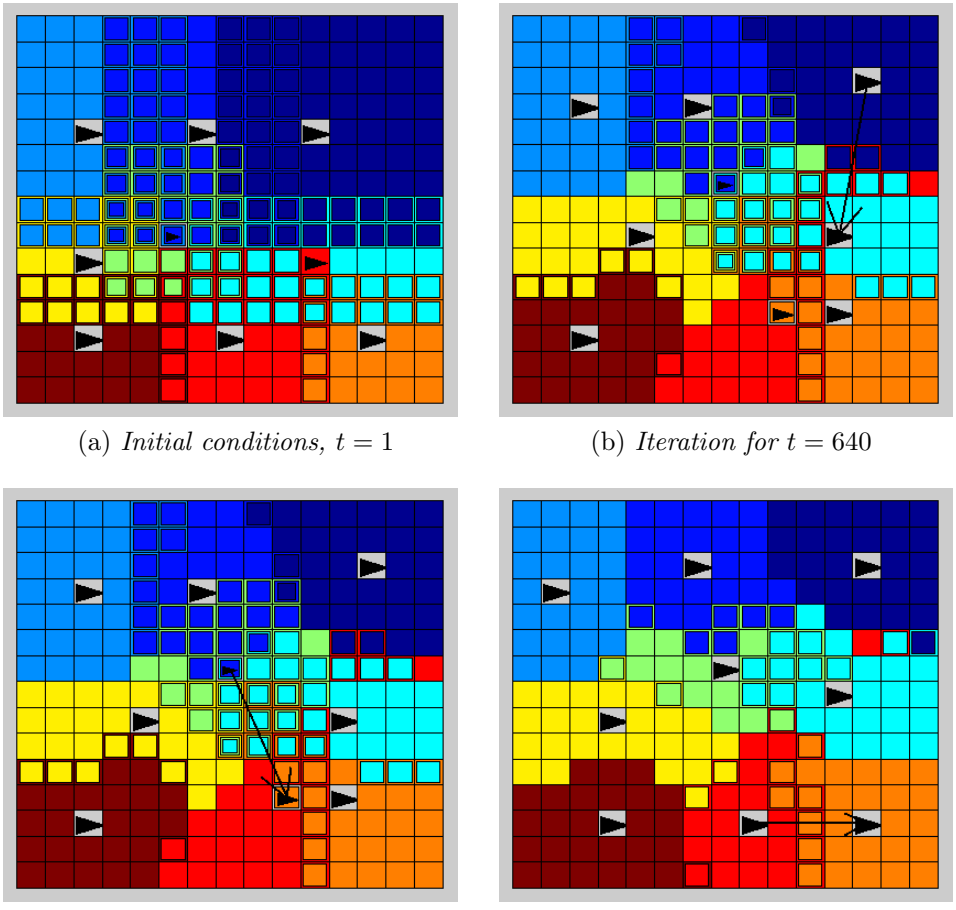
(b) *Plot of the mean of distance $\psi_i(t)$ between perimeter and centroids for all cameras*

Figure 27: Maximum error $\varepsilon(t)$ and index $\psi_i(t)$ for an environment patrolled by 4 agents with an unbalanced initialization.

we simulated the algorithm deleting in the code the part where the check of connection is performed. In Figure 28a we see the initial condition. It is unbalanced and without physical constraints, a favourable situation for disconnections (that are quite uncommon with a balanced and constrained initialization). In the iterations that follows the red camera spread up to the darkest blue one. In Figure 28b there is the situation right before the disconnection. The red camera receives informations from the green one and has to release some cells to reach an equitable partition. Cells that can be released are in the overlap and disconnect the red area as in Figure 28c. From that point the red area start to communicate with the darkest-blue one although their centroids are not 'neighbours'. Having a disconnection has poor meaning in real application and has to be avoided in every step of the algorithm -it might mean that they are not reachable-. In Figure 28d we see that disconnection keeps to present in futures instants.

6.4 Fault of a camera

Also with this kind of communication if a camera is busy or has a fault other cameras are able to spread their subregions to compensate the lack of one agent. The result is similar to the previous case with symmetric communication and its reported in Figure 29 and after, in Figure 30, there are the progress of the indexes.



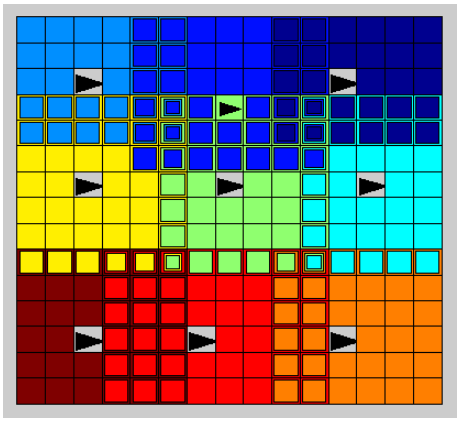
(a) *Initial conditions,* $t = 1$

(b) *Iteration for* $t = 640$

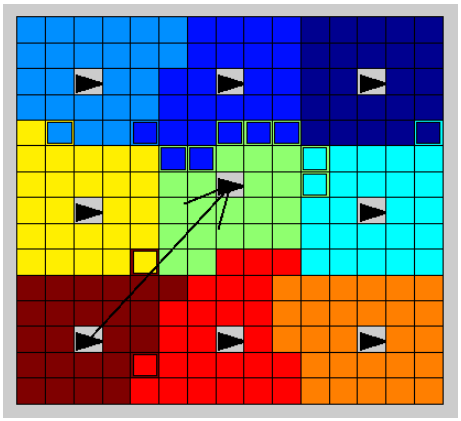
(c) *Iteration for* $t = 641$

(d) *Iteration for* $t = 900$

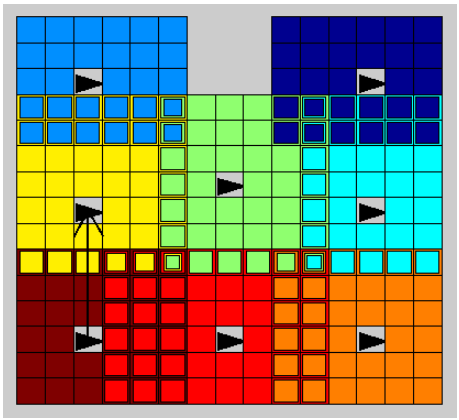
Figure 28: Some passages of the algorithm that doesn't perform a check on connection.



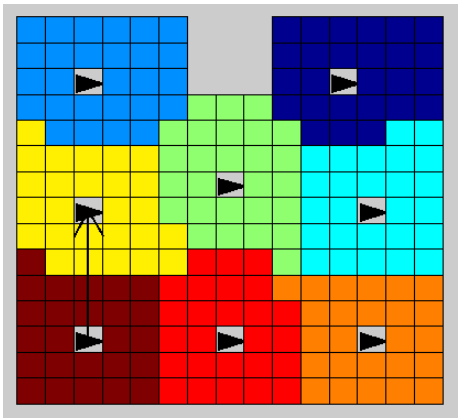
(a) *Initial condition,* $t = 1$



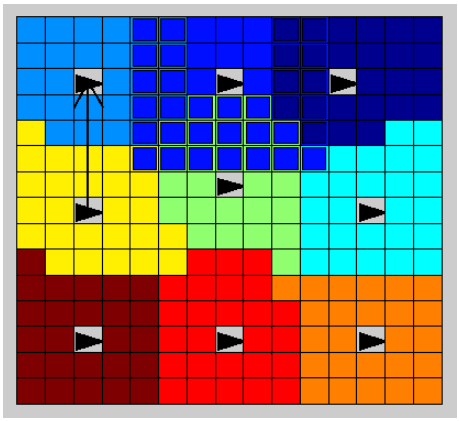
(b) *Configuration just before a camera fault,* $t = 300$



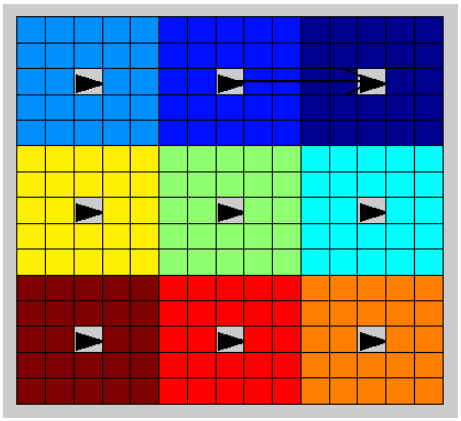
(c) *Reinitialization cause of the fault of the middle-blue camera,* $t = 301$



(d) *Final result with a camera missing,* $t = 1400$

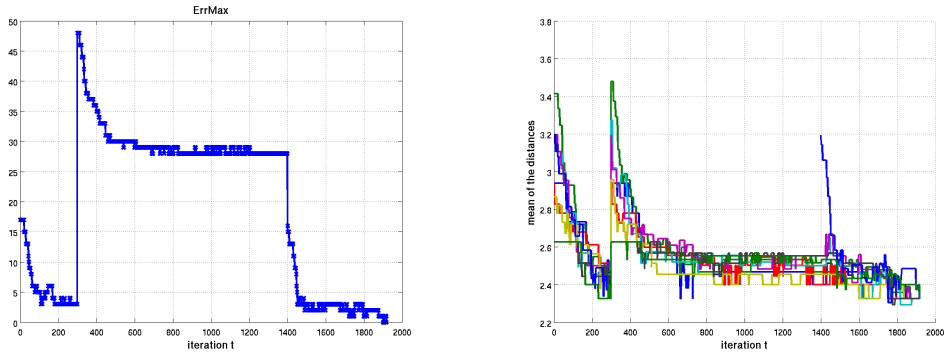


(e) *The middle-blue camera reappears spreading its physical constrain ,* $t = 1401$



(f) *Final result with all cameras,* $t = 1919$

Figure 29: Main passages of the algorithm when a fault of a camera appears.



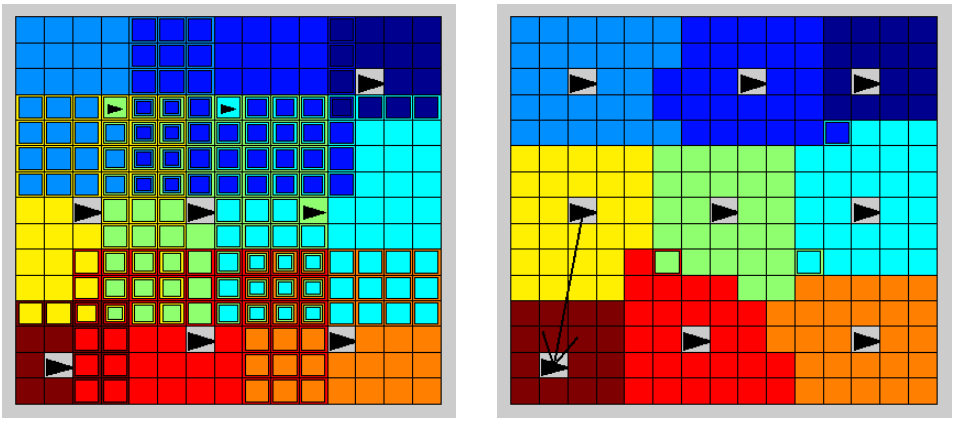
(a) *Index $\varepsilon(t)$, evident are points where a camera fault $t = 350$ and when it begins to work again $t = 1600$*

(b) *Progress of indexes $\psi_i(t)$*

Figure 30: An example where physical constraints don't contain the optimal solution (see brown and dark-blue areas).

6.5 Small physical constraints

In the previous examples an optimal final configurations is always achievable but there could be situation where this is not true due to physical constraints of some agents. In this case the algorithm has a behaviour similar to the solution developed for a symmetric communication. Agents more constrained spread their area as much as they can and don't change from that position. What left is covered and partitioned among the others. The only difference in this case is that oscillations near the equitable (among of course cameras that are not fixed) could be persistent, i.e. if it's not possible to partition the area left all with the same area size and the best border shape they try to modify this situation. Fortunately this is a matter of one point (eventually) for each camera and the global result is that the difference among the biggest and smallest area oscillate near the optimal result. The shape of subregions is kept as regular as possible. All this result can be seen in Figure 32a where $\varepsilon(t)$ keeps mainly the value of 13, and sometimes the optimal 12 and rarely increases to 14. Whereas in Figure 32c is shown indexes $\phi_i(t)$ that keep oscillate around the value of 2.5 and finally the view of areas sizes at the last instant in Figure 32b. In Figure 31 there is the initialization and the final configuration.



(a) *Initialization of cameras with their physical constraints*

(b) *Final result*

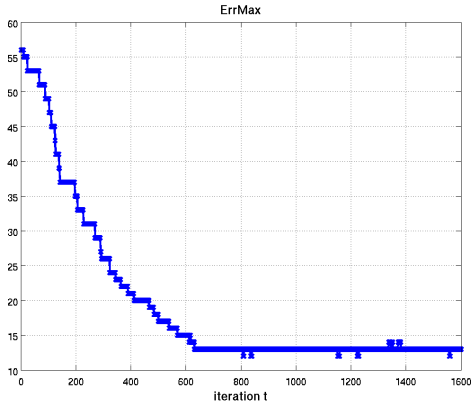
Figure 31: An example where physical constraints don't contain the optimal solution (see brown and dark-blue areas).

6.6 Statistical results

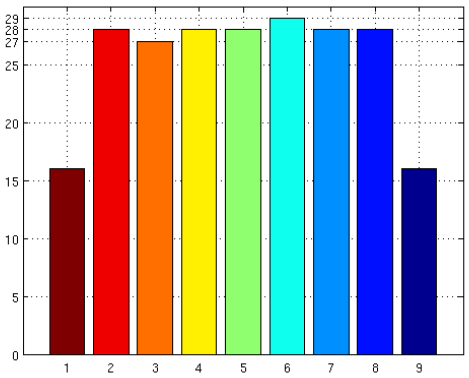
A statistical result of the behaviour of the partitioning strategies is generated running 500 times the algorithm on the square environment (discretized, as before, with 15 cells per edge) and a random initialization for each agents. In this test we don't consider physical constraints as we are more interested in seen the ability of reaching the optimal solution. As before physical constraints (if they contain the optimal configuration) could help the dynamics bounding the maximum spread of each camera (that, in a few cases, may became too irregular). Running this test we obtained that the optimal configuration is reached in 282 runs (so the 56.4%) with on average 2370 iterations (and a standard deviation of 858). For the other results the index $\eta(T)$ have mean and standard deviation:

$$\mathbb{E}[\eta] = 0.163$$
$$\sigma^2(\eta) = 0.0954$$

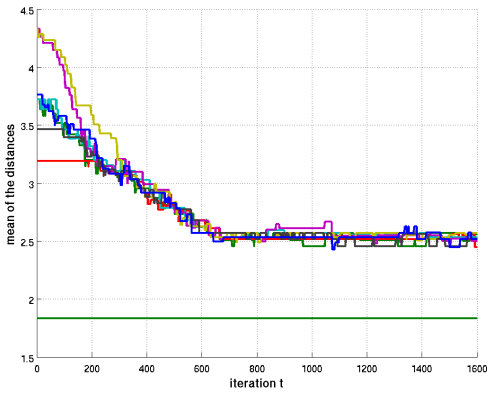
and looking at Figure 33 we see that just in few cases we don't obtain an equitable partition with a good shape for some cameras (remember that this result is made catching the worst sub-region of the final partition).



(a) *Evolving of the index $\varepsilon(t)$, it settles to the final value of* 12



(b) *Size of the area assigned to each camera in the final configuration*



(c) *Index $\psi_i(t)$ for $i = 1, \ldots, N$ the two more constrained cameras stay fixed, the others try to settle themselves to the same shape*;

Figure 32: An example where physical constraints don't contain the optimal solution (see brown and dark-blue areas).

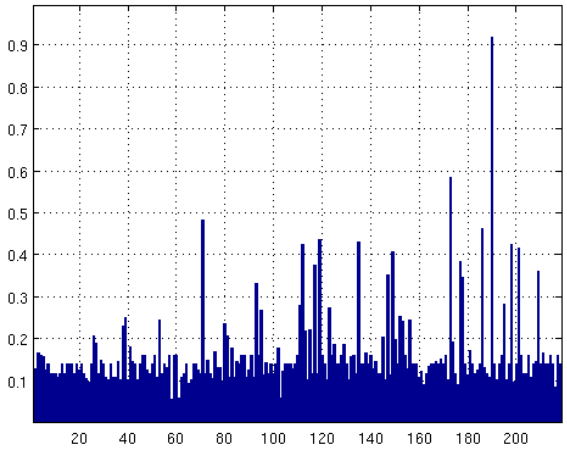


Figure 33: Difference η between the worst ψ among all cameras and ψ_{opt}

7 Conclusion

In the problem of patrolling a perimeter we encountered a distributed solution that equally distribute the segment to patrol to each agents. Respects to precedent results physical constraints of cameras are kept into account. The protocol of communication used is the pairwise gossip asymmetric, little demanding and suitable for large scaled cameras network, where even a synchronous communication may be too onerous. We encountered also a centralized solution modelling the problem as a linear programming problem and we solved it using the simplex algorithm. We simulated the algorithm proposed -in MATLAB code- and, comparing the two results -the centralized optimal solution and the distributed one-, we can affirm that the algorithm proposed converge to the optimal solution, i.e. minimize the length of the longest segment assigned to a camera. Moreover it equalize, as long as it is possible, all other segment sharing as best as it's possible the workload assigned to each agent. We tested the algorithm also in case of faulting cameras -an event that can happen in networks with a large amount of agents-. In a case like that we obtain an equal re-distribution of the workload among all the others agents.

In the second part we considered a generalization for the two dimensional, discrete case. The objective in here is double. Not only achieving an equitable partition with pairwise gossip communication but also achieving good shaped sub-regions. We developed two heuristic, one that works with a symmetric communication and the other for asymmetric communication. Simulative results shown that in both cases an equitable partition is always reached. The shapes optimality is not always achieved -although most of the times it is- but in general final results are composed with good shaped sub-regions. This improve and make easier the part of the patrolling, that can be made with well-knows techniques. In case of the fault of a camera all the others agents compensate, as long as it's possible, for the part of surface that is released by the faulting cameras and tries naturally to equalize their areas and to regularize their borders. In future works a good improvement could be done assuring the convexity of shapes in the transitory, and testing the algorithm in a non-convex scenario computing distances on a graph.

References

[1] Frederick R Adler and Deborah M Gordon. "Optimization, conflict, and nonoverlapping foraging ranges in ants." In: *The American naturalist* 162.5 (2003), pp. 529–543. URL: http://www.scopus.com/inward/record.url?eid=2-s2.0-0345581425&partnerID=40&md5=8c2a0dfb0b738108c58f049d8cc80467.

[2] Stephen Boyd et al. "Randomized gossip algorithms". In: *IEEE/ACM Trans. Netw.* 14 (SI 2006), pp. 2508–2530. ISSN: 1063-6692.

[3] F. Bullo, J. Cortes, and S. Martinez. *Distributed Control of Robotic Networks*. Applied Mathematics Series. Available at http://www.coordinationbook.info. 2009. ISBN: 978-0-691-14195-4.

[4] C.H.Caicedo-Nunez and M.Zefran. "Performing coverage on nonconvex domains". In: *IEEE Conf. on Control Application* (2008), pp. 1091–1024.

[5] J. W. Durham et al. "Discrete Partitioning and Coverage Control for Gossiping Robots". In: (Nov. 2010). to appear.

[6] Edward Fiorelli et al. "Multi-AUV control and adaptive sampling in Monterey Bay". In: *IEEE Journal of Oceanic Engineering*. 2004, pp. 935–948.

[7] Per-Olof Fjällström. "Algorithms for Graph Partitioning: A Survey". In: *International Symposium on Robotics Research (ISRR)* 3 ().

[8] A K Jain, M N Murty, and P. J. Flynn. *Data Clustering: A Review*. 1999.

[9] David Kempe, Alin Dobra, and Johannes Gehrke. "Gossip-Based Computation of Aggregate Information". In: *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*. FOCS '03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 482–. ISBN: 0-7695-2040-5. URL: http://dl.acm.org/citation.cfm?id=946243.946317.

[10] R.C.Mesquita L.C.A.Pimenta V.Kumar and G.A.Pereira. "Sensing and Coverage for a network of heterogeneous robots". In: *IEEE Conf. on Decision and Control* (2008), pp. 3947–3952.

[11] J. B. MacQueen. "Some Methods for Classification and Analysis of MultiVariate Observations". In: *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*. Ed. by L. M. Le Cam and J. Neyman. Vol. 1. University of California Press, 1967, pp. 281–297.

[12] Anna Mosser and Craig Packer. "Group territoriality and the benefits of sociality in the African lion, Panthera leo". In: *Animal Behaviour* 78.2 (2009), pp. 359–370. URL: http://www.sciencedirect.com/science/article/B6W9W-4WKS6J1-3/2/52ca094fd25b6be653a508dd7a7414d5.

[13] and J.J.Slotine M.Shwager D.Rus. "Decentralized, adaptive coverage control for networked robots". In: *International Journal of Robotics Research* vol.28 (2007), pp. 357–375.

[14] M.Zhong and C.G.Cassandras. "Distributed coverage control in sensor network environments with polygonal obstacles". In: *IFAC World Congress* (2008), pp. 4162–4167.

[15] D.Krass O.Baron O.Bermann and Q. Wang. "The equitable location problem on the plane". In: *European Journal of Operational Research* (2007).

[16] M. Pavone. "Dynamic Vehicle Routing for Robotic Networks". PhD thesis. Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, 2010.

[17] M. Pavone, E. Frazzoli, and F. Bullo. "Distributed policies for equitable partitioning: Theory and applications". In: 2008, pp. 4191–4197.

[18] M. Pavone et al. "Equitable partitioning policies for robotic networks". In: Kobe, Japan 2009, pp. 2356–2361.

[19] S.P.Lloyd. "Least square quantization in PCM". In: *IEEE Transaction on Information Technology* 28.2 (1982), pp. 129–137.

[20] Vijay V. Vazirani. *Approximation Algorithms*. Springer, 2004.

[21] Mac; Rus Yun Seungkook; Schwager and Daniela L. "Coordinating Construction of Truss Structures using Distributed Equal-mass Partitioning". In: *International Symposium on Robotics Research (ISRR)* vol.28 (2009), pp. 357–375.